**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

# APLICAÇÃO DE ABORDAGENS PROBABILÍSTICAS EM DISPOSITIVOS MÓVEIS PARA A SEGURANÇA DE PEDESTRES

Eduardo Massao Kobayashi

FLORIANÓPOLIS – SC
2018/1

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**
**CENTRO TECNOLÓGICO**
**DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**
**CURSO DE SISTEMAS DE INFORMAÇÃO**

# APLICAÇÃO DE ABORDAGENS PROBABILÍSTICAS EM DISPOSITIVOS MÓVEIS PARA A SEGURANÇA DE PEDESTRES

Eduardo Massao Kobayashi

Orientador: Prof. Dr. Elder Rizzon Santos

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

FLORIANÓPOLIS – SC
2018/1

Eduardo Massao Kobayashi

# APLICAÇÃO DE ABORDAGENS PROBABILÍSTICAS EM DISPOSITIVOS MÓVEIS PARA A SEGURANÇA DE PEDESTRES

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

_____

Prof. Dr. Cristian Koliver

Coordenador do Curso

Banca Examinadora:

_____

Prof. Dr. Elder Rizzon Santos

Orientador

_____

Profª. Drª. Luciana de Oliveira Rech

_____

Prof. Dr. Ricardo Azambuja Silveira

Dedico à minha filha Sarah, nascida enquanto desenvolvia este trabalho.
Deu um novo sentido à minha vida.

# AGRADECIMENTOS

Agradeço a Deus primeiramente, pois sou uma pessoa religiosa. Sei que isso não é muito popular hoje em dia, mas acredito que há alguém maior que nós, sempre disposto a nos ajudar e nos guiar na jornada da vida.

Agradeço em especial à minha esposa Suyara, pela paciência e compreensão nesses longos anos de estudos. Muitas vezes não pude dar atenção e dedicar tempo de qualidade que ela merecia, mas ainda assim sempre me apoiou.

Agradeço à minha família, que consiste de mãe e irmãos, por seu incentivo para que eu investisse em uma educação superior, e para que não desistisse.

Agradeço também ao meu orientador, professor Elder, pela maneira como me orientou, sempre de maneira respeitosa, assim como pela dedicação e paciência.

# RESUMO

Estudos apontam para o aumento de acidentes em ambientes urbanos devido ao uso intenso de dispositivos móveis, que tiram a atenção do usuário. Como parte de um projeto maior, que busca desenvolver um modelo computacional que possibilite medir o nível de percepção do usuário, este trabalho se destina a desenvolver um aplicativo para dispositivo móvel que utilize uma rede bayesiana para representar a consciência situacional. As redes bayesianas se propõem a resolver problemas que envolvem incerteza, possuindo uma representação visual que mostram a relação de causa entre as variáveis, o que é adequado com a temática estudada. O aplicativo foi desenvolvido na plataforma Android, tendo como premissa que o algoritmo de inferência bayesiana fosse executado dentro da aplicação, utilizando-se para isso a biblioteca do UnbBayes. A adaptação dessa biblioteca para uso em dispositivos Android apresenta-se como uma contribuição secundária deste trabalho. Foram realizados testes com a aplicação e o resultado é apresentado.

**Palavras-chave:** inteligência artificial, redes bayesianas, consciência situacional, aplicativo para dispositivos móveis, Android, UnbBayes

# ABSTRACT

Studies point to the increase of accidents in urban environments due to the intense use of mobile devices, which take the attention of the user. As part of a larger project that seeks to develop a computational model to measure the level of user perception, this paper is intended to develop a mobile application that uses a Bayesian network to represent situational awareness. Bayesian networks propose to solve problems involving uncertainty, having a visual representation that show the causal relationship between the variables, which is adequate with the subject studied. The application was developed in the Android platform, assuming that the Bayesian inference algorithm was executed within the application, using the UnbBayes library. The adaptation of this library for use on Android devices is a secondary contribution of this work. Tests were performed with the application and the result is presented.



**Keywords**: artificial intelligence, Bayesian networks, situational awareness, mobile app, Android, UnbBayes

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIAÇÕES

**API** Application Programming Interface (Interface de Programação de Aplicações)

**CS** Consciência Situacional

**EM** Expectation-Maximization

**IA** Inteligência Artificial

**IDE** Integrated Development Environment (Ambiente de Desenvolvimento Integrado)

**MVC** Model-View-Controller

**RB** Rede Bayesiana

# SUMÁRIO

# 1 INTRODUÇÃO

O uso de dispositivos móveis tem aumentado gradativamente, sendo um fenômeno global. Mas seu uso perto de estradas e ferrovias tem sido motivo de preocupação devido ao também crescente número de acidentes e fatalidades que vêm acontecendo devido ao uso desatento do dispositivo. O que se percebe é que o uso intenso do aparelho causa a distração cognitiva do usuário, que não percebe as eventuais situações de risco no local em que se encontra. A partir disso, surgiu um projeto que se propõe a desenvolver um modelo computacional de consciência situacional para usuários de dispositivos móveis (smartphones, tablets, etc.) quando perto de estradas e ferrovias, intitulado Road Awareness (SANTOS; FAZENDA, 2016).

Este trabalho pretende contribuir para esse projeto, aplicando aprendizado de máquina para a classificação dos dados de entrada, que permita chegar a conclusões com relação ao nível de CS do usuário de dispositivo móvel.

O aprendizado de máquina é uma área da Inteligência Artificial (IA) dedicada ao desenvolvimento de algoritmos e técnicas que permitam ao computador aprender, isto é, que permitam ao computador melhorar seu desempenho em alguma tarefa de forma automática. Existem várias abordagens para se implementar aprendizado de máquina (e diversos algoritmos), dentre elas podemos citar as redes neurais artificiais, as redes bayesianas, computação evolucionária (algoritmos genéticos), modelos de Markov e outros (LUGER, 2016). Para este trabalho propõe-se o uso de redes bayesianas.

Uma rede bayesiana (RB) é um modelo probabilístico, na forma de um grafo acíclico dirigido, construída com nodos, representando variáveis aleatórias discretas ou contínuas, e arcos direcionados, representando as relações entre elas. Os nodos filhos são dependentes de seus pais, e possuem uma probabilidade associada. As RB’s são construídas usando conhecimento de especialistas ou usando algoritmos eficientes que aprendem a rede (BUCZAK, 2016).

Dependendo da aplicação, a rede pode ser usada para explicar a interação entre as variáveis ou para calcular um resultado provável com base nos dados de entrada. As tabelas de probabilidade podem ser calculadas a partir dos dados de treinamento disponíveis. Entre os principais objetivos para o treinamento de RB’s

estão inferir variáveis não observadas, a aprendizagem de parâmetros e a aprendizagem da estrutura (BUCZAK, 2016).

Para que o aplicativo conseguisse chegar a uma conclusão quanto ao nível de consciência do usuário, poderia-se realizar a inferência bayesiana em um servidor remoto ou de dentro da própria aplicação. Para o presente trabalho procurou-se seguir essa última alternativa, adaptando e fazendo uso de uma API desenvolvida na linguagem Java mas adaptada para ser executada em uma aplicação Android.

## 1.1 OBJETIVOS

### 1.1.1 Objetivo Geral

O objetivo geral é implementar um aplicativo para dispositivo móvel para a segurança de pedestres, fazendo uso de uma RB para representar a consciência situacional e analisando o uso do aparelho com dados dos sensores.

### 1.1.2 Objetivos Específicos

1. Analisar os métodos e algoritmos relacionados à RB's;
2. Definir a RB para representar a consciência situacional;
3. Selecionar e utilizar uma API para realizar a inferência bayesiana no dispositivo;
4. Analisar quais sensores do smartphone são relacionados ao problema;
5. Implementar um aplicativo Android para a segurança de pedestres;
6. Realizar testes.

## 1.2 MOTIVAÇÃO

O que motivou este trabalho foi a vontade de criar uma ferramenta que ajudasse os deficientes visuais a se contextualizarem-se com o ambiente, pelo que seria desenvolvido um aplicativo para dispositivos móveis que fizesse a análise do ambiente através de sensores, como câmera e microfone, classificando pessoas e objetos, e fazendo conclusões quanto ao lugar em que o indivíduo se encontrasse. Seria necessário a utilização de aprendizado de máquina para implementar a classificação.

Tratando deste tema com o orientador, o mesmo comentou a existência de um projeto em andamento com outra universidade, intitulado “Computational Model of Situational Awareness for users of smartphones in the vicinity of traffic”, que trata da percepção do ambiente e desatenção por parte do usuário de dispositivos móveis, com foco em situações em que o colocariam em perigo, sendo que a utilização de aprendizado de máquina nesse caso seria muito semelhante com a ideia do aplicativo para deficientes visuais. Dessa forma, surgiu a oportunidade de utilizar os dados coletados no projeto para o desenvolvimento de um modelo de aprendizado de máquina, o que pouparia o trabalho de coletar dados, mas que implicaria em desenvolver um modelo voltado para o projeto. Optou-se por seguir esse caminho. Apesar disso, o modelo ainda pode ser aproveitado para ambas as ideias.

Com o modelo desenvolvido seria possível utilizá-lo em um aplicativo para dispositivos móveis conforme descrito no início desta motivação.

# 2 FUNDAMENTAÇÃO TEÓRICA

## 2.1 CONSCIÊNCIA SITUACIONAL

Endsley (1995) define consciência situacional (CS) como "a percepção de elementos do ambiente considerando tempo e espaço, a compreensão de seu significado e a projeção de seu status em um futuro próximo". Também pode ser considerada um campo de estudo preocupado em compreender os ambientes em que é crítica a tomada de decisão, como acontece em áreas complexas e dinâmicas como aviação, controle de tráfego aéreo e operação de grandes e complexos sistemas, ou mesmo para tarefas mais comuns do dia-a-dia como caminhar e dirigir.

Segundo Endsley (1995), o primeiro passo para se obter a CS é perceber as características e a dinâmica de elementos relevantes no ambiente. Um motorista de carro, por exemplo, precisa perceber outros veículos e pedestres próximos, a sinalização das ruas, assim como perceber as características e dinâmica de seu próprio carro. O nível seguinte de consciência, que seria o nível 2, é alcançado quando se tem a compreensão da atual situação, o que significa que não basta estar ciente dos elementos, precisa-se compreender o significado deles com base em objetivos pertinentes. O motorista de carro então precisa entender o significado das informações apresentadas no painel de seu carro e presentes na sinalização de outros carros e das ruas, que o auxiliarão na tomada de decisão. Por fim, um nível maior de CS é alcançado quando se consegue fazer projeções futuras dos elementos do ambiente, ou seja, quando se consegue antecipar o que vai acontecer. Assim, o motorista precisa identificar antecipadamente possíveis situações de colisão para pode evitá-las.

Dessa forma, alguém com um bom nível de CS tem grande chance de tomar decisões corretas em sistemas dinâmicos e complexos. Além disso, o estudo e entendimento dos componentes que constituem a CS possibilitam o desenvolvimento de interfaces mais eficientes e de melhores programas de suporte à decisão (ENDSLEY, 1995).

O presente trabalho tem como foco o impacto que o uso de dispositivos móveis tem na CS. O uso de smartphone afeta a percepção do usuário, e ao tentar utilizar esses dispositivos enquanto realiza alguma outra tarefa, como dirigir, andar de bicicleta ou mesmo caminhar perto de rodovias e ferrovias, têm grande

probabilidade de causar acidentes. O uso desses dispositivos afeta a atenção do usuário em dois níveis: oclusão, e cegueira e surdez por desatenção.

A oclusão ocorre quando o usuário não consegue realmente ver ou escutar outros elementos do ambiente enquanto está utilizando o dispositivo. Pode ser visualmente quando ao fixar o olhar na tela não conseguir perceber veículos se aproximando ou que deixou a calçada e está agora na rua, ou pode ser auditiva quando ao usar fones de ouvidos não conseguir perceber os sons de motor e buzinas de carrro. A cegueira e surdez por desatenção acontece quando o usuário, mesmo vendo e escutando outros elementos do ambiente, os ignora devido a sua atenção estar voltada exclusivamente para o dispositivo, mesmo que esses elementos externos coloquem em risco a sua segurança (SANTOS; FAZENDA, 2016).

## 2.2 APRENDIZADO DE MÁQUINA

Lobato (2016) define o aprendizado de máquina como “a habilidade de um computador aprender sem ter sido explicitamente programado para resolver determinada tarefa específica”, sendo o resultado da utilização de algoritmos eficientes que conseguem encontrar padrões a partir de dados disponíveis.

Segundo Buczak (2016), o aprendizado de máquina geralmente consiste de duas fases: treinamento e testes, em que são realizados os seguintes passos:

- Identificar características nos dados;
- Identificar um subconjunto dos atributos necessários para a classificação;
- Aprender o modelo usando dados de treinamento;
- Usar o modelo treinado para classificar os dados desconhecidos.

Para Lobato (2016), o aprendizado de máquina é realizado por meio das etapas observadas na Figura 1:

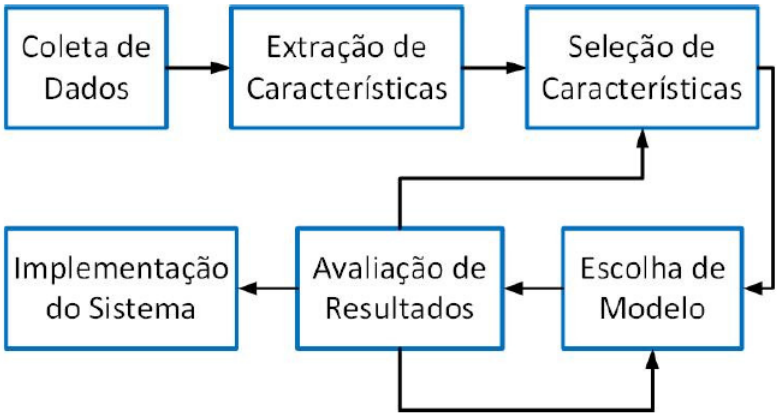


Figura 1: Etapas da aplicação de aprendizado de máquina. Fonte: Lobato (2016)

A primeira etapa consiste na coleta de dados e pode ser feita por meio de um banco de dados existente, por meio da captura de dados através de sensores e salvos na base de dados, ou pela captura de fluxos de dados em tempo real. A próxima etapa é a extração de características, extraindo-se diretamente da base de dados ou através da combinação de campos de dados. Após extrair o máximo de características, faz-se a seleção daquelas que melhor se relacionam com o problema que se quer resolver. A etapa seguinte é a escolha de um modelo adequado, que consiste na utilização de um algoritmo eficiente, seguido da etapa de testes, em que o modelo é avaliado quanto à sua eficácia. Caso o resultado não seja satisfatório, pode-se escolher outro modelo ou retornar para a seleção de características, finalizando, por fim, com a implantação do sistema (LOBATO, 2016).

O aprendizado de máquina pode ser dividido, de maneira geral, em aprendizado supervisionado e não supervisionado. No aprendizado supervisionado, os dados de treino são etiquetados e o objetivo é encontrar uma relação ou modelo que explique os dados. No aprendizado não supervisionado não há etiquetação dos dados e o objetivo nesse caso é encontrar padrões ou conhecimento nos dados (LOBATO, 2016).

## 2.3 REDES BAYESIANAS

De acordo com Charniak (1991), existem alguns tipos de problemas que queremos que os sistemas resolvam, mas que são difíceis de solucioná-los somente pela lógica. São problemas que envolvem incerteza. Uma determinada doença, por exemplo, possui diversos sintomas e é difícil muitas vezes chegar a um diagnóstico

preciso somente através desses sintomas. Tosse, febre e calafrios são sintomas que podem indicar gripe, mas também estão presentes em outras doenças. Esse “pode” indica uma possibilidade, o que traz a ideia de probabilidade. Isso é o que as RB’s se propõe a fazer, utilizar a teoria de probabilidades para resolver problemas em que não se consegue ser conclusivo através de lógica.

As RB’s surgiram da necessidade de se analisar um grande número de variáveis e o impacto que cada uma tem sobre as demais (NEAPOLITAN apud ARA-SOUZA, 2010). Também podem ser consideradas como uma representação visual e informativa do conjunto de probabilidades das variáveis envolvidas no problema (ARA-SOUZA, 2010).

Uma RB pode ser definida como uma rede probabilística, constituída de uma estrutura associada a uma distribuição de probabilidades. Essa estrutura é representada por meio de grafos direcionados, conectados e acíclicos, compostos de nodos e arcos. Os nodos são círculos e correspondem às variáveis aleatórias, e os arcos são setas e representam a dependência probabilística direta entre duas variáveis (BUCZAK, 2016).


Figura 2 - Representação de nodos e arcos. Fonte: Ara-Souza (2010)

A estrutura pode ser de conexões simples, aqui incluídas a árvore simples e poli árvore, no qual o grafo possui apenas um único caminho ligando dois nodos, sem levar em conta a direção, ou pode ser de múltiplas conexões. A tabela de probabilidade condicional se apresenta como um elemento importante da estrutura (ARA-SOUZA, 2010). Por fazerem uso de uma estrutura gráfica, as RB’s permitem que se perceba as relações de causa entre as variáveis, sendo sua leitura bastante intuitiva, além de facilitar a interpretação dos resultados (SARABANDO, 2010).

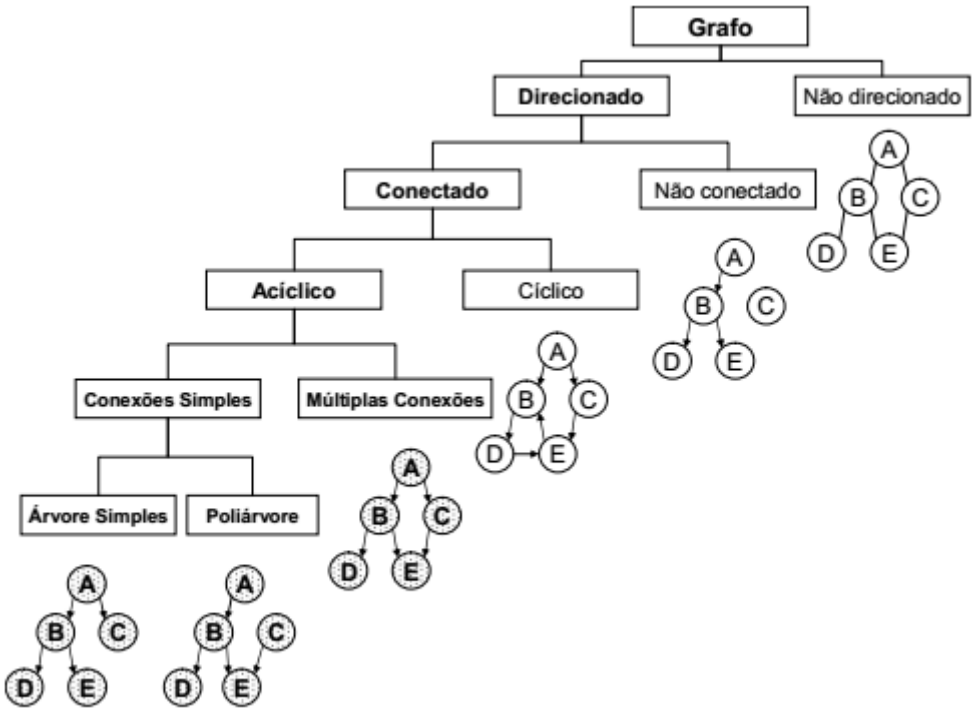


Figura 3 - Descrição de tipos de grafos e suas estruturas. Fonte: Ara-Souza (2010)

Para especificar a distribuição das probabilidades de uma RB, é necessário definir as probabilidades de todos os nodos-raízes (nodos que não possuem predecessor) e as probabilidades condicionais de todos os demais nodos, com base na relação direta com seu predecessor (CHARNIAK, 1991). Assim, cada variável da rede pode tomar um valor dentro de um intervalo finito e nelas estão definidas as probabilidades condicionais. A probabilidade condicional é a probabilidade de determinado evento X ocorrer sabendo que o evento Y ocorreu, dada por P(X|Y). O Teorema de Bayes relaciona as probabilidades dos dois acontecimentos X e Y com as suas probabilidades condicionais mútuas, dado por:

$$P(X|Y) = P(X|Y)\frac{P(X)}{P(Y)}$$

Quando uma variável tem valor conhecido e inserido na rede, temos o que se chama de evidência, sendo utilizada para realizar inferência. Inferência é um termo que se refere a atualização das probabilidades da rede com base nas evidências, ou seja, uma vez inserido um valor conhecido na rede, as probabilidades precisam ser ajustadas (ARA-SOUZA, 2010). As redes bayesianas nos permitem calcular as probabilidades condicionais mesmo que tenhamos os valores de apenas alguns nodos. A inferência pode ser difícil de ser realizada dependendo da complexidade da

estrutura e da quantidade de variáveis, e dependendo das particularidades da rede e do algoritmo utilizado, o tempo necessário para o cálculo pode ser extremamente grande (NP-hard) (CHARNIAK, 1991).

Para estruturas mais comuns existem dois tipos de algoritmos utilizados para realizar inferência: algoritmos exatos, com maior precisão nos resultados mas que exige um maior processamento, e algoritmos aproximados, com menor precisão mas que exigem um menor processamento. Os algoritmos para solução exata devem ser aplicados às redes de conexão simples, o que torna o cálculo mais simples. Em uma rede com múltiplas conexões, o cálculo é mais complexo. Na situação mostrada na Figura 4, por exemplo, quando se quer saber a probabilidade condicional de c, uma alteração no valor de d afeta não somente c, mas também b, que por sua vez altera o valor de a, afetando c (CHARNIAK, 1991).

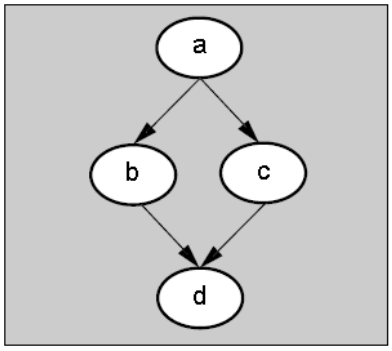


Figura 4 - Múltipla conexão. Fonte: Charniak (1991)

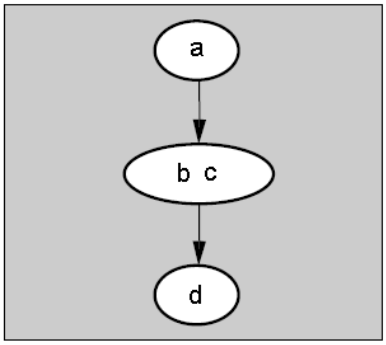


Figura 5 - Conexão simples obtida por meio de clustering. Fonte: Charniak (1991)

Os algoritmos de inferência exata baseiam-se na repetição do teorema de Bayes considerando a estrutura de independência entre determinados conjuntos de variáveis, e são indicados quando a quantidade de variáveis é inferior a 36. Dentre os algoritmos frequentemente citados na literatura temos: passagem de mensagens e método de formação de agrupamentos. O algoritmo de passagem de mensagens é aplicado em estruturas do tipo árvore (simples ou poli árvore). O método de formação de agrupamentos tem como objetivo transformar uma estrutura de múltiplas conexões, que exigem um esforço computacional muito grande, em uma estrutura do tipo árvore, utilizando a técnica de clustering, o que é possível observar na Figura 5 (ARA-SOUZA, 2010).

Os algoritmos de inferência aproximada baseiam-se em processos de

simulação. Alguns dos algoritmos frequentemente citados na literatura são: amostragem por rejeição, ponderação de probabilidade e Gibbs Sampling. A ideia do algoritmo de amostragem por rejeição é gerar n amostras, rejeitando aquelas que não contém determinada evidência. Já a ponderação de probabilidade evita o descarte, uma vez que gera apenas amostras que possuem a evidência dada. A ideia básica do algoritmo de Gibbs Sampling é gerar cada evento a partir do estado atual da rede, e a partir daí, gerar o próximo estado por uma passagem aleatória dependente desse estado atual (ARA-SOUZA, 2010).

Pode-se construir uma RB manualmente, através do conhecimento obtido por meio de um especialista e da literatura, ou pode-se aplicar técnicas de aprendizagem para a estimação dos parâmetros e da estrutura, através de algoritmos aplicados sobre a base de dados (SARABANDO, 2010).

A estimação de parâmetros refere-se à estimação das probabilidades condicionais para cada variável-nodo da rede, podendo ser realizada para conjunto de dados completos e incompletos. Para dados incompletos pode-se utilizar o algoritmo EM (utilizado pelo Hugin), que se baseia nos casos observados para estimar os valores faltantes. A estimação também pode ser realizada utilizando estimadores de máxima verossimilhança e estimadores bayesianos. A estimação de estrutura refere-se à melhor disposição de dependências e independências entre as variáveis que explique melhor o problema em estudo, podendo fazer uso de busca heurística e do conceito de independência condicional dos atributos da rede, ou uma abordagem híbrida, para realizar a estimação (ARA-SOUZA, 2010).

## 2.4 IMPLEMENTAÇÃO DE REDE BAYESIANA EM DISPOSITIVOS MÓVEIS

Os dois principais sistemas operacionais utilizados em dispositivos móveis são o IOS da Apple e o Android da Google. Também vem tentando conquistar espaço o Windows Phone da Microsoft. Essas empresas têm investido bastante em aplicações que utilizam IA. A Apple lançou a Siri, a Google lançou o Google Assistant e a Microsoft a Cortana, que fazem com que o smartphone seja uma ferramenta muito inteligente, bastando solicitar através de comando de voz o que queremos, como por exemplo, inicializar aplicativos, obter a previsão do tempo, resolver problemas matemáticos, pesquisar lugares, entre outros. Muito da IA é utilizada com o objetivo de entender os gostos e interesses do usuário para poder prover melhores

sugestões e propiciar uma melhor experiência com o aparelho. E há muitas outras aplicações, como identificação de objetos e rostos em fotos, sugestões de respostas em chats, etc.

Muitas soluções em IA realizam o processamento em servidores remotos, sendo que o smartphone apenas realiza as requisições. Mas há muito o que vem sendo feito para possibilitar o processamento no próprio dispositivo, principalmente para as situações em que não há conexão com a internet, ou que acarrete um custo para se processar remotamente. A aplicação de aprendizado de máquina internamente no dispositivo também é limitado. O treinamento do modelo, por exemplo, não é realizado no dispositivo. A maioria das soluções em IA para dispositivos móveis utilizam redes neurais. Tensorflow[1], por exemplo. RB's não são muito utilizadas. Poucos artigos comentam, e não há tutoriais na internet com passo a passo de como montar uma em dispositivos móveis.

Para o trabalho a intenção é utilizar a plataforma Android, devido já ter experiência no desenvolvimento de aplicativos dessa plataforma, além de possuir um smartphone com o sistema Android. Para a implementação de RB em dispositivos móveis, poderiam ser seguidas 3 abordagens:

1. Utilizar uma API Java open source como biblioteca no Android, como Jayes[2] ou UnBBayes, considerando a compatibilidade entre as plataformas;
2. Utilizar alguma solução python como código nativo no Android;
3. Utilizar alguma biblioteca do Tensorflow, pois possui muitos avanços, apesar do foco em redes neurais;
4. Implementar a RB utilizando a API do Android, tendo como base o código de alguma API já desenvolvida, podendo ser na linguagem Java, considerando que a plataforma Android é baseada na núcleo do Linux e utiliza a linguagem Java para o desenvolvimento dos aplicativos.

---

1 "TensorFlow." https://www.tensorflow.org/mobile.

2 "Jayes Studio." https://jayesstudio.com/.

## 2.5 FERRAMENTAS

### 2.5.1 Hugin

O Hugin Lite[3] é a versão gratuita e limitada do software. Com o software é possível construir uma RB entrando no modo de edição, podendo adicionar variáveis discretas e contínuas, e os arcos. Para cada variável é possível inserir as probabilidades. Uma das mais importantes habilidades que o software possui é a de realizar a inferência, bastando iniciar o processo de compilação da rede para que as probabilidades sejam propagadas.

### 2.5.2 Android Studio

O Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos Android e é baseado no IntelliJ IDEA. Além do editor de código e das ferramentas de desenvolvedor do IntelliJ, o Android Studio oferece ainda recursos na criação de aplicativos Android, como emulador de dispositivo e compatibilidade com código em linguagem nativa.

3"HUGIN-lite | Hugin Expert." https://www.hugin.com/index.php/hugin-lite/.

# 3 TRABALHOS RELACIONADOS

Neste capítulo foram selecionados alguns temas relacionados com o estudo aqui desenvolvido, apresentando trabalhos correlatos.

## 3.1 CONSTRUÇÃO DE REDES BAYESIANAS

Sarabando (2010) propõe em seu trabalho um estudo do comportamento de RB's no prognóstico da sobrevivência do câncer de próstata. É introduzido de modo claro e objetivo os conceitos relacionados às RB's, salientando as principais vantagens, como a interpretação dos resultados por meio da visualização dos grafos e a possibilidade de se conjugar conhecimento científico com dados estatísticos de forma simples. Comenta-se cada uma das ferramentas que foram utilizadas para a construção da rede.

O trabalho faz um comparativo entre as diferentes maneiras de se construir RB's, seja por meio do auxílio de um especialista ou utilizando softwares que aprendem a rede. Inicialmente é criada a estrutura com o conhecimento de um especialista comparando-se com a literatura da área, descrevendo-se cada uma das variáveis utilizadas, inclusive com a discretização quando necessário. A especificação das probabilidades é feita por meio de cálculos para as probabilidades a priori e com a ajuda de um programa para as probabilidades condicionais, com base em uma base de dados disponível. Em um segundo momento são criadas diferentes redes em diferentes softwares, utilizando-se para isso algoritmos de aprendizado.

## 3.2 USO DE REDES BAYESIANAS EM DISPOSITIVOS MÓVEIS

Lee e Cho (2012) propõem um sistema de gerenciamento de energia para smartphone Android usando RB. O sistema em questão faz a coleta de dados, como uso de CPU e RAM, proximidade, aceleração, uso da bateria, entre outros, a fim de identificar o comportamento e uso das funções do aparelho. Após um pré-processamento de dados visando as discretização, os mesmos são submetidos à inferência bayesiana com o objetivo de determinar a probabilidade de se desligar ou não alguma funcionalidade.

Yi e Cho (2012) propõem um sistema semelhante, mas o foco é economizar

energia diminuindo o uso do GPS, um tipo de sensor que consome muita energia. A ideia é coletar dados de sensores do smartphone que consomem menos, como acelerômetro e bússola, e utilizar um método usando inferência bayesiana para obter informações de localização mais eficientemente, evitando o uso desnecessário do GPS.

## 3.3 DISTRAÇÃO PROVOCADA PELO USO DE SMARTPHONES

Em seu artigo, Schwebel et al. (2012) estuda o impacto que a distração causada pelo uso de smartphone tem nos pedestres, mais especificamente conversar ao telefone, encaminhar mensagens e escutar música, quando próximos de rodovias. Para melhor compreender o comportamento dos pedestres, foi realizado um experimento em que 138 participantes foram imersos em um ambiente semi-virtual. O experimento consistia em atravessar uma rua em quatro diferentes situações: (1) escutando música, (2) conversando no celular, (3) encaminhando mensagens de textos e (4) sem nenhuma distração.

Os resultados demonstraram que aqueles que escutavam música ou que encaminhavam mensagens estavam mais propensos a serem atropelados.

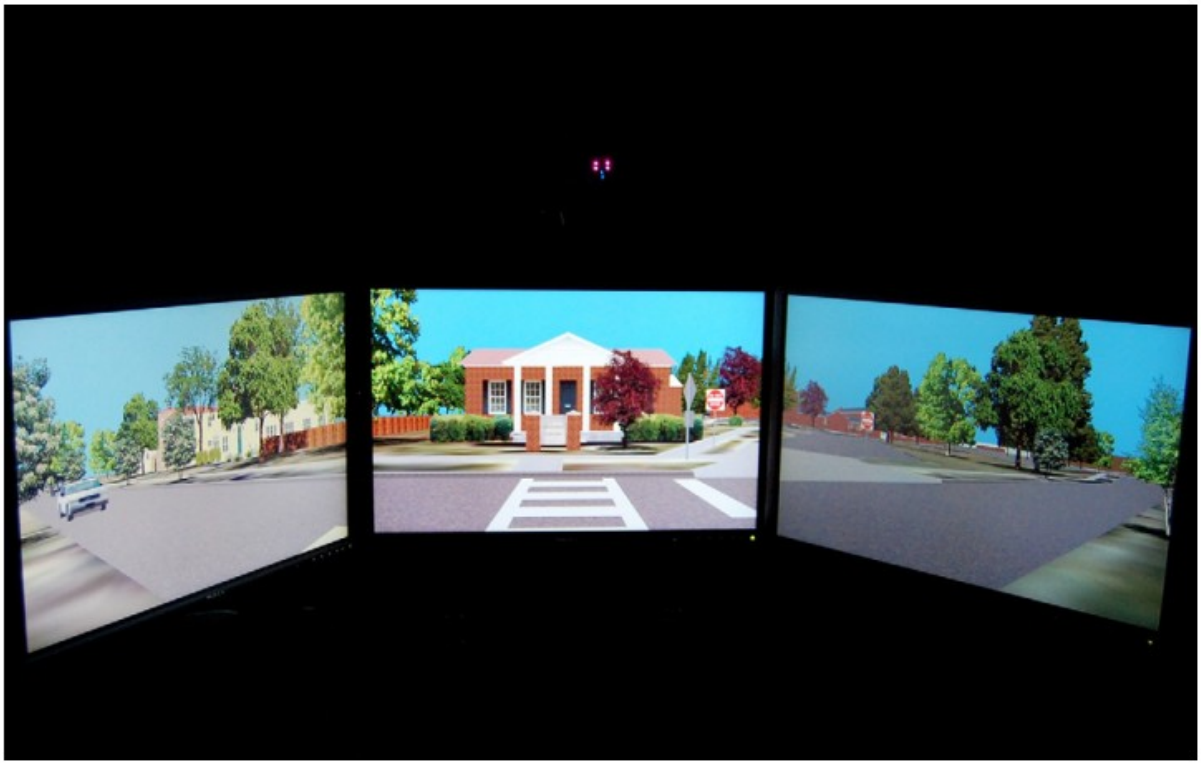

Figura 6 – Ambiente virtual. Fonte: Schwebel et al. (2012)

# 4 USO DA REDE BAYESIANA PARA CONSCIÊNCIA SITUACIONAL EM DISPOSITIVOS MÓVEIS

Neste capítulo é apresentado o desenvolvimento do aplicativo, iniciando-se pelo estudo de caso e descrevendo-se o sistema proposto. É explicado o uso de RB e como foi possível executar um algoritmo de inferência bayesiana dentro da aplicação. O aplicativo é construído na plataforma Android e ao final os testes são realizados.

## 4.1 ESTUDO DE CASO

O projeto Road Awareness se propõe a desenvolver um modelo computacional de CS voltado para usuários de dispositivos móveis que utilizam seus aparelhos próximos de rodovias e ferrovias, com o objetivo de melhor compreender como esses usuários se comportam nesse ambiente e em que situações sua segurança é comprometida. Através dos dados obtidos e do estudo realizado seria possível melhorar a segurança dos pedestres (SANTOS; FAZENDA, 2016).

O projeto pretende caracterizar os níveis de deficiência (oclusão e cegueira por desatenção) quando os dispositivos móveis estão sendo usados e combiná-los com o perfil de comportamento do usuário e recursos coletados pelo próprio dispositivo para construir modelos computacionais que prevejam o nível de CS de um usuário (SANTOS; FAZENDA, 2016).

O projeto visa responder às seguintes questões (SANTOS; FAZENDA, 2016):

- Como a direção do olhar, a oclusão auditiva e o mascaramento, a atividade no dispositivo e a cegueira não-intencional prejudicam a percepção do usuário?
- O nível de CS pode ser previsto, em uso, a partir de recursos derivados de sensores disponíveis em um dispositivo móvel moderno?
- O conhecimento dos riscos do ambiente pode ser combinado com o nível previsto de CS para fornecer alertas?
- Esse sistema é eficiente para fornecer segurança aos usuários, complementando os sentidos que foram desviados para o dispositivo?

O projeto faz uso de uma plataforma de realidade virtual 3D, localizada na Universidade de Salford, Reino Unido, em que o usuário é imerso em um ambiente urbano simulado.

O ambiente simula uma cidade urbana, contendo vias em que circulam carros, possuindo uma faixa de pedestres em que circulam pequenos animais. Junto à faixa de pedestres fica um semáforo que estabelece a passagem desses animais quando está verde e bloqueando-os quando fica vermelho. Os animais presentes na simulação eventualmente atravessam a faixa, mas ignoram se vem ou não carros pela via. O experimento consiste em um usuário ficar próximo da faixa, sendo responsável por controlar o semáforo por meio de um smartphone, devendo deixar o sinal verde assim que for seguro atravessar pela faixa para que os animais possam fazer isso sem que sejam atropelados. O experimento considerou ainda a possibilidade de utilizar distrações para tirar a atenção do usuário. Assim, em meio às várias atividades que precisa executar, é analisada a CS do usuário.

Figura 7 – Usuário imerso no ambiente da simulação. Fonte: Pereira (2017)

## 4.2 SISTEMA PROPOSTO

O sistema proposto consiste de três componentes: uma entrada com dados de sensores, um processamento interno e uma saída em forma de alerta ao usuário.

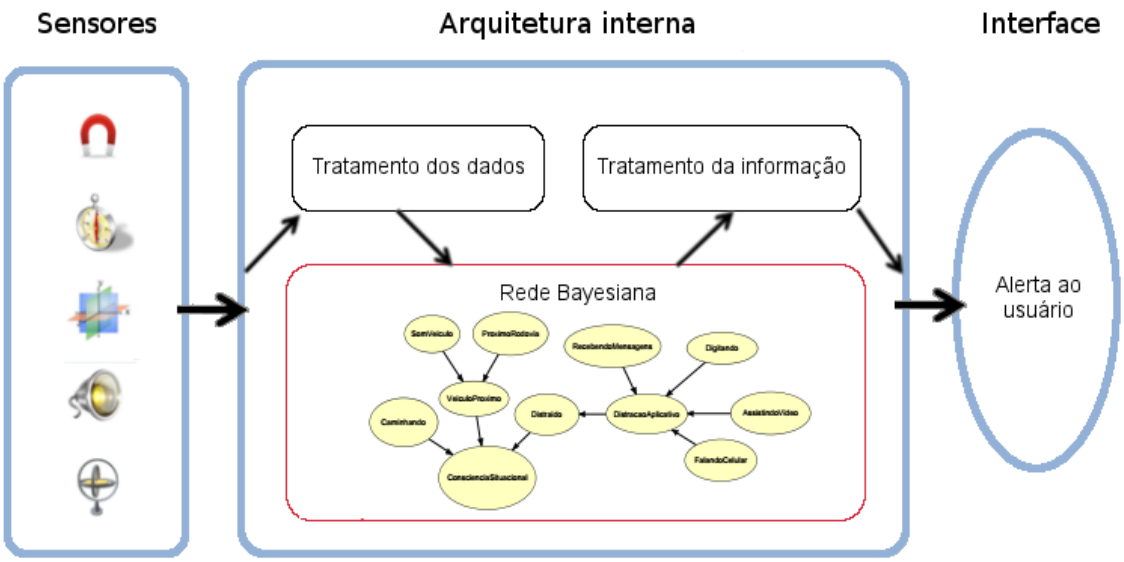


Figura 8 - Diagrama do sistema proposto. Fonte: Do autor (2017)

Através da API Android é possível obter dados dos seguintes sensores: Acelerômetro, Temperatura Ambiente, Gravidade, Giroscópio, Luz, Aceleração Linear, Campo Magnético, Orientação, Pressão, Proximidade, Umidade Relativa, Vetor de Rotação e GPS. Para o problema proposto não será necessário utilizar todos os sensores, pois fogem do contexto. Segue uma tabela com os sensores que estão relacionados com o problema, mais especificamente em identificar se o usuário do smartphone está caminhando e se está próximo de vias urbanas:

| **Sensor** | **Descrição** | **Usos Comuns** |
|---|---|---|
| Acelerômetro | Mede a força de aceleração em $m/s^2$ que é aplicada a um dispositivo nos três eixos físicos (x, y e z), incluindo a força da gravidade. | Detecção de movimento (agitação, inclinação, etc.). |
| Campo Magnético | Mede o campo geomagnético do ambiente para os três eixos físicos (x, y, z) em μT. | Criação de bússola. |

| | | |
|---|---|---|
| Orientação | Mede os graus de rotação que um dispositivo faz em torno dos três eixos físicos (x, y, z). | Determinação da posição do dispositivo. |
| Proximidade | Mede a proximidade de um objeto em cm em relação à tela de um dispositivo. Este sensor normalmente é usado para determinar se o aparelho está sendo mantido próximo ao ouvido de uma pessoa. | Posição do telefone durante uma chamada. |
| Vetor de Rotação | Mede a orientação de um dispositivo fornecendo os três elementos do vetor de rotação do dispositivo. | Detecção de movimento e detecção de rotação. |

Tabela 1 - Sensores relacionados com o problema.

Fonte: Da documentação Android[4] (adaptado)

Além desses sensores, através da API do Android também é possível identificar algumas ações do usuário, como ouvir música, se ele está digitando, recebendo mensagens ou falando ao celular.

O processamento interno visa identificar o nível de consciência do usuário, e para isso é utilizada uma RB. Dessa forma, a medida que os dados são obtidos através dos sensores, é realizado o tratamento desses dados para possibilitar a inserção das evidências na rede. O resultado dessa inferência bayesiana representa o nível de CS do usuário em relação ao ambiente em que se encontra, e essa informação é tratada para então ser emitido um alerta ao usuário.

A aplicação executa em background, sempre monitorando o ambiente e a interação do usuário com o dispositivo, e uma vez identificado um nível baixo de CS, seja pelo uso intenso do dispositivo ou pela identificação de um ambiente mais crítico, ou a composição de ambos, a aplicação emite alertas ao usuário, seja por meio de avisos sonoros, visuais ou por vibração.

4"Sensors Overview | Android Developers." https://developer.android.com/guide/topics/sensors/sensors_overview. Acessado em 29 mai. 2018.

## 4.3 CONSTRUÇÃO DA REDE BAYESIANA

Pereira (2017), em seu trabalho de conclusão de curso, também utiliza como estudo de caso o projeto Awareness. Em seu trabalho ele constrói uma RB com o fim de desenvolver e experimentar uma técnica de aprendizagem por reforço com o propósito de aprender os parâmetros da rede.

Para a construção da RB, ele utilizou os dados coletados através de um experimento realizado no ambiente virtual, conforme estudo de caso, em que participaram 20 usuários. Além de ter que executar as atividades do simulador, os participantes foram submetidos a três diferentes situações: (1) o participante deveria apenas apertar o botão indicando estar consciente com relação ao carro; (2) ele deveria, além de apertar o botão indicando consciência, também responder perguntas no smartphone; e (3) ele deveria, além de apertar o botão indicando consciência e responder as perguntas, usar ainda fones de ouvido. No simulador os carros poderiam vir de quatro direções, frente, atrás, direita e esquerda, e poderiam vir com ou sem som. O tempo que o participante levou para indicar a consciência também foi considerado. Conseguindo executar as tarefas com sucesso, o participante era considerado como consciente. Na tabela seguir é possível observar parte dos dados obtidos:

| id | Consciente | Simulacao | SomCarro | DirecaoCarro | Tempo |
|---|---|---|---|---|---|
| 6 | Sim | Fones | Não | Esquerda | 12.055s |
| 6 | Não | Fones | Sim | Esquerda | 16.260s |
| 6 | Sim | Perguntas | Não | Atrás | 1.383s |
| 6 | Não | Perguntas | Não | Esquerda | 11.582s |
| ... | ... | ... | ... | ... | ... |

Tabela 2 - Parte do conjunto de dados utilizado. Fonte: Pereira (2017)

Os dados foram tratados para que pudessem ser utilizados para a construção da rede. Resumidamente, os passos seguidos foram os seguintes:

1. Dos 962 registros, 40 deles foram descartados por estarem incompletos;
2. Os dados da variável Simulação foram classificados como Sim, para a ocorrência de Fones e Perguntas, e Nao para a ocorrência de Botão, e o nome da variável foi renomeado para DistracaoApp;

3. Para a variável Tempo, Pereira (2017) faz algumas considerações com relação ao processamento do simulador, em especial quanto a geração de carros, que impactaram na medição do tempo que um participante levava para indicar consciência, o que seria a sua percepção, o que acarretou na necessidade de se fazer um tratamento mais elaborado dos dados. Assim, foi necessário primeiro calcular a média do tempo considerando a ocorrência de som do carro relativa a sua direção, gerando os dados da Tabela 3. Após, somente para os casos de Consciente, os dados foram classificados como Presente, se o tempo estava abaixo ou igual a média, e Tardia se o tempo estava acima da média. Para os casos de Inconsciente, os dados foram classificados como Ausente. O nome da variável foi renomeado para Percepcao.

| Som \ Direcao | Trás | Frente | Esquerda | Direita |
|---|---|---|---|---|
| Sim | 6.279 | 7.069 | 10.635 | 8.297 |
| Não | 8.518 | 8.160 | 11.629 | 9.646 |

Tabela 3 - Média de tempo de Consciente para direções e sons dos carros.
. Fonte: Pereira (2017)

Com base nos dados, a estrutura da RB é construída de forma manual. Após a aplicação do aprendizado por reforço, obteve-se a rede mostrada na figura:

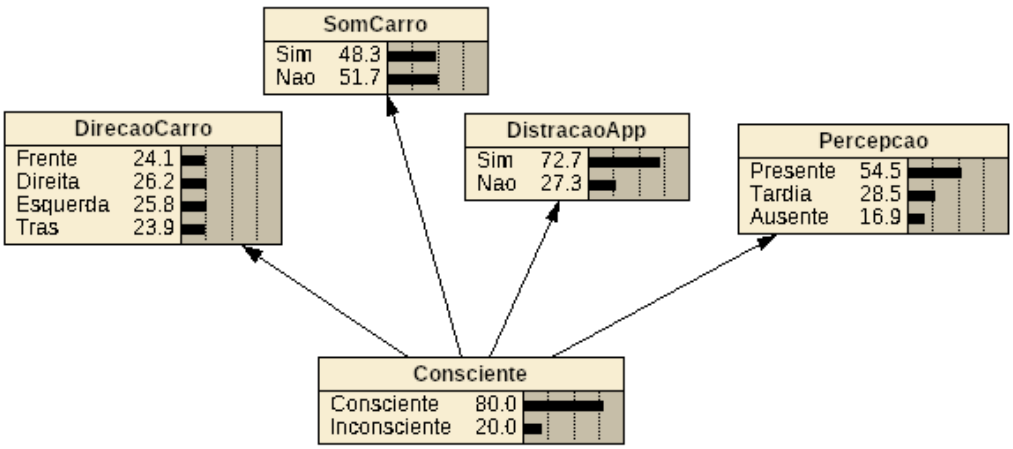


Figura 9 - RB após aplicação do aprendizado por reforço. Fonte: Pereira (2017)

### 4.3.1 Estrutura e Especificação das Probabilidades

Para a criação da estrutura é necessário definir quais nodos compõe a rede, estabelecendo as variáveis correspondentes a cada nodo. Para a especificação das probabilidades é possível, utilizando uma base de dados, calcular ou treinar através de algum algoritmo. Considerando que a intenção do presente trabalho é contribuir para o projeto Awareness, optou-se por utilizar os mesmos nodos com as mesmas variáveis da rede apresentadas por Pereira (2017) para a construção da RB, assim como a especificação das probabilidades.

Houve uma tentativa de inverter o fluxo da rede, por entender que faria mais sentido que o fluxo fosse em direção ao nodo Consciente, e por achar assim que a inserção de evidências nos demais nodos impactaria mais diretamente na consciência. Considerando que o fluxo da rede original seria alterado, não seria possível utilizar a especificação das probabilidades como foram definidas previamente, pois o que antes foi calculado como probabilidade a priori, passa a ser probabilidade condicional, e vice versa. Com base nos dados do projeto Awareness foi possível construir a rede da Figura 10, mas testes preliminares acabaram mostrando que a inserção de evidência no nodo DistracaoApp, ao invés de diminuir a consciência, aumentava. Dessa forma, essa nova rede não pôde ser aproveitada, sendo utilizada no aplicativo a rede apresentada por Pereira (2017).

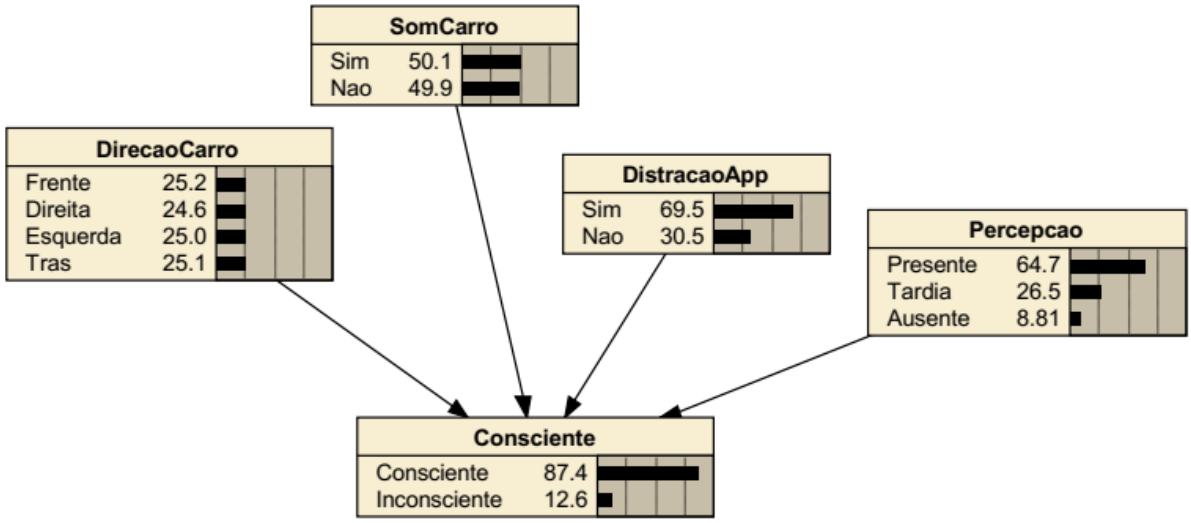


Figura 10 - RB com alteração do fluxo e nova especificação de probabilidade.

Fonte: Do autor (2018)

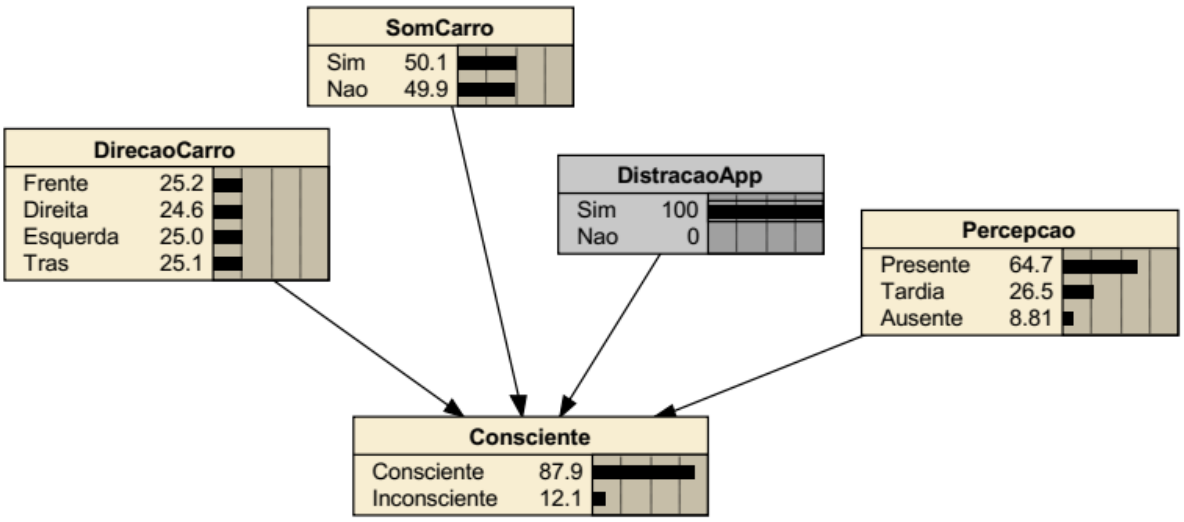


Figura 11 - RB com alteração do fluxo e evidência no DistracaoApp.

Fonte: Do autor (2018)

## 4.4 APLICATIVO PARA SEGURANÇA DE PEDESTRES

A fim de que o aplicativo pudesse inferir a CS, seria necessário que ele aceitasse a inserção de evidências e que executasse algum algoritmo de inferência bayesiana dentro do próprio dispositivo. Foram realizadas pesquisas na internet relacionadas à implementação de RB no sistema Android.

Pesquisando no Google por “android bayesian network”, dentre os resultados obtidos, um projeto se mostrou interessante denominado “bayes-android[5]”, porém o mesmo não possui documentação, e se descreve como um simples simulador de RB. Sem documentação talvez seria difícil adaptar para os propósitos deste trabalho.

Realizando pesquisas relacionadas à implementação de aprendizado de máquina, uma das soluções apresentadas foi a utilização do weka[6], descrito como uma coleção de algoritmos de aprendizado de máquina com foco em mineração de dados. Dessa forma, pesquisando no Google por “android weka”, foi encontrado o projeto “Weka-for-Android[7]”, que nada mais é do que o projeto open source do Weka sem as classes incompatíveis com a API do Android. A princípio seria uma solução

5"GitHub - txus/bayes-android: A Bayes network simulator for Android.", https://github.com/txus/bayes-android.

6"Weka 3 - Data Mining with Open Source Machine Learning Software ...." https://www.cs.waikato.ac.nz/ml/weka/.

7"GitHub - rjmarsan/Weka-for-Android: the Weka project with the GUI ...." https://github.com/rjmarsan/Weka-for-Android.

interessante, com uma documentação extensa, mas que seria necessário conhecer as ferramentas voltadas para RB.

Pensando no fato de que a plataforma Android tem alta compatibilidade com a plataforma Java, foram realizadas pesquisas de soluções desta última relacionadas com o desenvolvimento de RB's. Utilizar projetos Java como bibliotecas em projetos Android é algo bem comum, já previsto pela API do Android. Há várias soluções em Java para a construção de RB's, dentre as principais temos Weka, como já citado, Jayes, JavaBayes e UnBBayes.

Dentre as opções disponíveis, foi escolhido o UnbBayes, uma vez que essa ferramenta tem o foco em RB, e foi possível compreender bem o projeto através do artigo de Matsumoto (2014), o qual possui boas explicações e bons exemplos.

### 4.4.1 Adaptação do UnBBayes para Android

UnbBayes[8] é uma aplicação desenvolvida pelo grupo de Inteligência Artificial do departamento de Ciências da Computação da Universidade de Brasília, que tem como premissa possibilitar a construção de modelos gráficos probabilísticos e a realização de inferência. Possui interface gráfica e faz uso de plugins para estender suas funcionalidades (MATSUMOTO, 2014).

O projeto oferece um repositório que inclui plugins para rede Bayesiana (BN), Diagrama de Influência (ID), Rede Bayesiana de Seção Múltipla (MSBN), Rede Bayesiana Híbrida (HBN), Rede Bayesiana Orientada a Objetos (OOBN) Modelo Relacional Probabilístico (PRM), Multi-Entity Bayesian Network (MEBN), Probabilistic Web Ontology Language (PR-OWL), aprendizagem de parâmetros, aprendizagem de estrutura e aprendizado incremental de RB's, bem como amostragem de dados estatísticos, avaliação de desempenho de classificação, mineração de dados e vários outros algoritmos para inferência bayesiana. O projeto foi desenvolvido na linguagem Java e implementa como mecanismo padrão de inferência o algoritmo de junção de árvore (MATSUMOTO, 2014).

Para poder utilizar o UnbBayes na plataforma Android como uma biblioteca, foi necessário realizar algumas adaptações, uma vez que a API do Android, apesar de ser baseada em Java, não processa as classes Java de Interface Gráfica (Java

8"UnBBayes - The UnBBayes Site." http://unbbayes.sourceforge.net/. Acessado em 29 mai. 2018.

Swing, por exemplo), ou classes que utilizam hardware, uma vez que possui classes específicas. Portanto, foi necessário retirar do projeto essas classes.

Inicialmente, do código fonte do UnBBayes foram retiradas algumas funcionalidades para melhor entender o código, pois é bem extenso. Primeiramente, foram retiradas partes da interface gráfica, como as diversas opções do menu, restando somente o menu para abrir uma RB do arquivo, assim como as chamadas de métodos correspondentes. O passo seguinte foi retirar a funcionalidade de se gerar/carregar diagramas de influência (influence diagrams). Uma vez entendido o código, como próxima etapa optou-se por retirar o restante da interface gráfica, por conter as classes das bibliotecas awt e swing (pacote javax), não compatíveis com a API do Android.

Pelo mesmo motivo, foi necessário também retirar a funcionalidade de carregamento de plugins. Também foram retiradas a maioria das dependências de classes externas, como aquelas que tratam de logs, pois muitas delas eram incompatíveis com a API do Android. Os seguintes arquivos foram retirados do projeto original: commons-logging-1.0.4.jar, javahelp-2.0.02.jar, jaxme2-rt-0.5.1.jar, jaxmeapi-0.5.1.jar, jpf-1.5.jar, log4j-1.2.17.jar, xalan-2.7.0.jar e xml-apis-1.0.b2.jar.

Para utilizar uma biblioteca Java no Android, com o projeto aberto no Android Studio deve-se clicar em “File”, “Project Structure”, vai abrir uma janela, no item “Modules” clicar em “app”, depois em “Dependencies”. Depois é só clicar no botão “+” e importar um arquivo .jar.

O resultado é um projeto leve, com tamanho de 4,74 MB, e o arquivo JAR com tamanho de 1,87 MB, frente aos 48,5 MB do projeto original. Porém possui um único algoritmo de inferência, o de junção de árvore, além de perder várias funcionalidades.

### 4.4.2 Sensores

Importante esclarecer que o termo sensor, por definição, se refere aos dispositivos que fazem parte do hardware do smartphone que conseguem medir estímulos físicos específicos, como a bússola e o giroscópio. Neste trabalho, além do significado descrito anteriormente, o termo sensor também se refere à identificação, por meio do smartphone, de qualquer atividade relacionada ao uso do dispositivo.

Dentre os usos frequentes do smartphone, os mais comuns são jogar, assistir música e se comunicar com outra pessoa, seja por fala ou por meio de mensagens de texto. Considerando que o aplicativo será executado em segundo plano, uma vez que a ideia é analisar o uso do smartphone, houveram algumas restrições quanto ao que poderia ser monitorado.

A identificação de jogar, por exemplo, é complexa, uma vez que essa atividade faz uso intenso de diversos recursos do dispositivo. A identificação de toque na tela ou mesmo digitação de caracteres no teclado virtual atualmente é restrita à aplicação que executa essas atividades, não sendo acessíveis por aplicações externas. Em versões anteriores do Android era possível realizar esta identificação de fora da aplicação, mas isso foi bloqueado com versões recentes da API. Além disso, dependendo do modelo do smartphone, ele pode vir muitas vezes com apenas um único sensor, que geralmente é o acelerômetro, o que restringe bastante o desenvolvimento de uma aplicação que consiga abranger uma gama maior de dispositivos.

Portanto, foram testados somente a identificação de caminhada, ouvir som, chamada recebida/iniciada, e tela ligada.

Considerar a identificação da caminhada pelo usuário do smartphone foi uma questão bastante relevante no desenvolvimento do aplicativo, uma vez que quando se fala na segurança do pedestre, dificilmente ele estaria em perigo se estiver parado. Mas não chega a ser uma atividade que pudesse ser classificada como uma distração causada pelo uso do aparelho, e por isso não foi considerada como uma evidência que pudesse ser inserida na RB.

Com relação a identificação de tela ligada, a probabilidade de que o usuário esteja utilizando o smartphone e distraído é muito grande, mas colocá-la como evidência na rede implicaria que só o fato de ligar o dispositivo o aplicativo classificaria a consciência como baixa, independente de qualquer outra distração. Dessa forma, ela não foi considerada como uma evidência a ser inserida na rede. Outra situação é que o usuário poderia estar distraído mesmo com a tela desligada, como ocorreria na oclusão auditiva ao usar fones de ouvido.

Ouvir música e estar com uma ligação telefônica ativa são atividades que realmente distraem o usuário, e assim foram consideradas como evidências na rede.

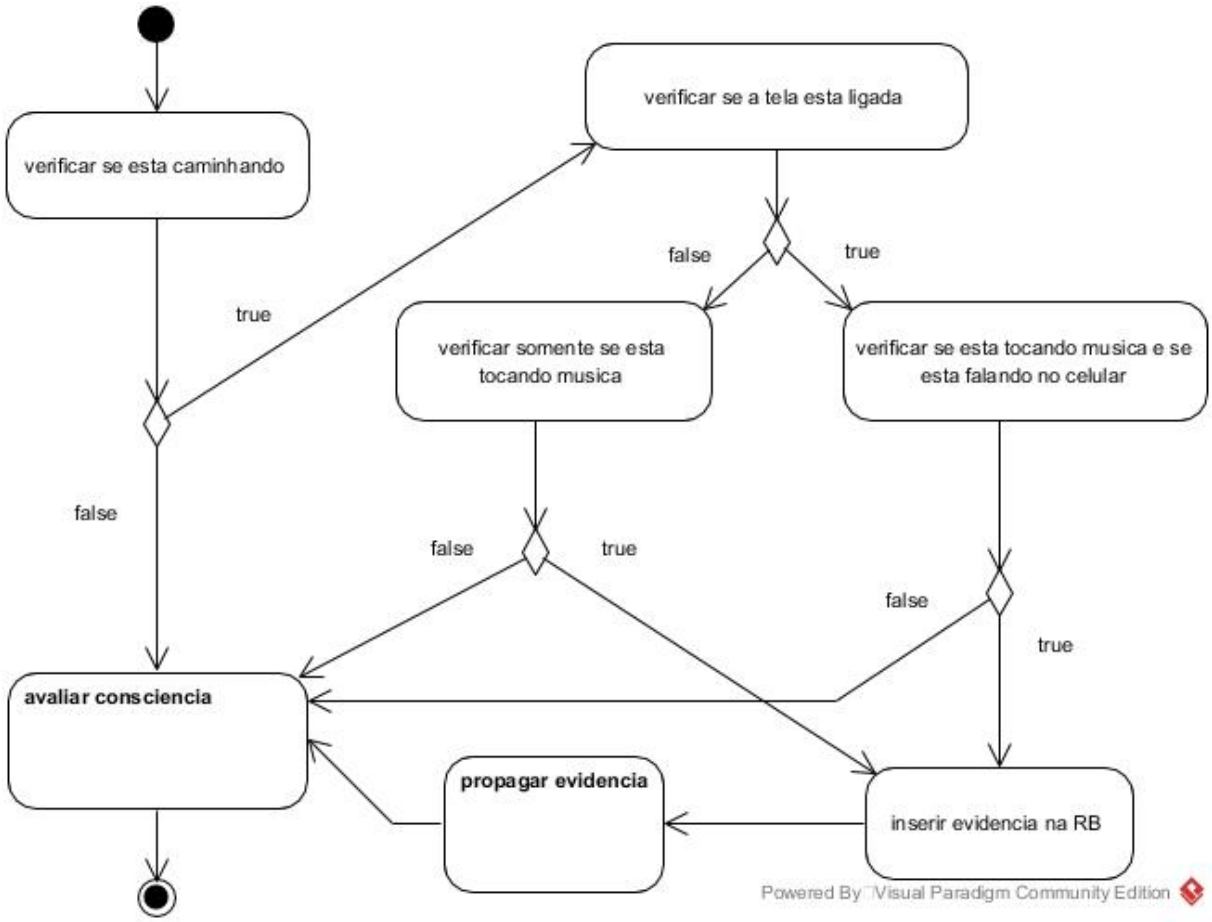


Figura 12 - Diagrama de atividades do tratamento de dados para inserção na RB.
Fonte: Do autor (2018)

Dessa forma, o tratamento dos dados dos sensores dentro da aplicação, para eventual inserção de evidência na RB, é feito conforme o diagrama da Figura 12:

1. Primeiramente é analisado se o usuário está caminhando, sendo esta uma condição necessária para possível inserção de evidência. Se confirmada a ocorrência de caminhada, passa-se para uma segunda análise. Senão, passa-se para o final, onde é verificado o nível de consciência;
2. Em um segundo momento, havendo caminhada, é feita a verificação de tela ligada, o que apenas altera a verificação dos demais dados dos sensores;
3. Se a tela não está ligada, é verificado apenas se há algum som sendo reproduzido, considerando a situação de escutar música com a tela desligada;

4. Se a tela está ligada, é verificada se há a ocorrência de ligação telefônica ou se algum som está sendo reproduzido. Uma chamada telefônica é identificada após se apertar o botão de telefone, tanto para chamadas efetuadas quanto para chamadas recebidas. Os sons aqui identificados podem ser tanto de reprodução de música quanto originados por players de vídeo;
5. Havendo a identificação de som ou chamada telefônica, é feita a inserção da evidência na RB e propagada para o restante da rede, finalizando com a avaliação do nível de consciência. Não havendo, passa-se para o final, onde é verificado o nível de consciência.

O modelo desenvolvido no projeto Awareness e utilizado para a construção da RB também trouxe algumas limitações quanto à inserção de evidências na rede. Com relação ao nodo DirecaoCarro, não haveria como o smartphone identificar a direção do carro (esquerda, frente, etc.) em relação ao usuário/aparelho. Também com relação ao nodo Percepcao, seria difícil identificar se a reação do usuário é tardia, presente ou ausente. Na verdade, o propósito do aplicativo é justamente suprir a ausência da percepção. Somente com relação à identificação de som do carro para o nodo SomCarro, acredita-se que haveria a possibilidade de se implementar, pois o tratamento e identificação de sons não é algo novo, mas isso aumentaria demais a complexidade do trabalho, e por isso foi deixado de lado.

### 4.4.3 Implementação do Aplicativo

O aplicativo foi implementado na plataforma Android, utilizando a IDE Android Studio versão 3.1.2. Procurou-se estruturar o projeto de modo a deixar separado a interface gráfica do domínio do problema, sendo a comunicação entre ambas realizada por uma classe intermediária, conforme é proposto pelo padrão MVC. Dessa forma, as classes foram separadas em três pacotes: view, control e model.

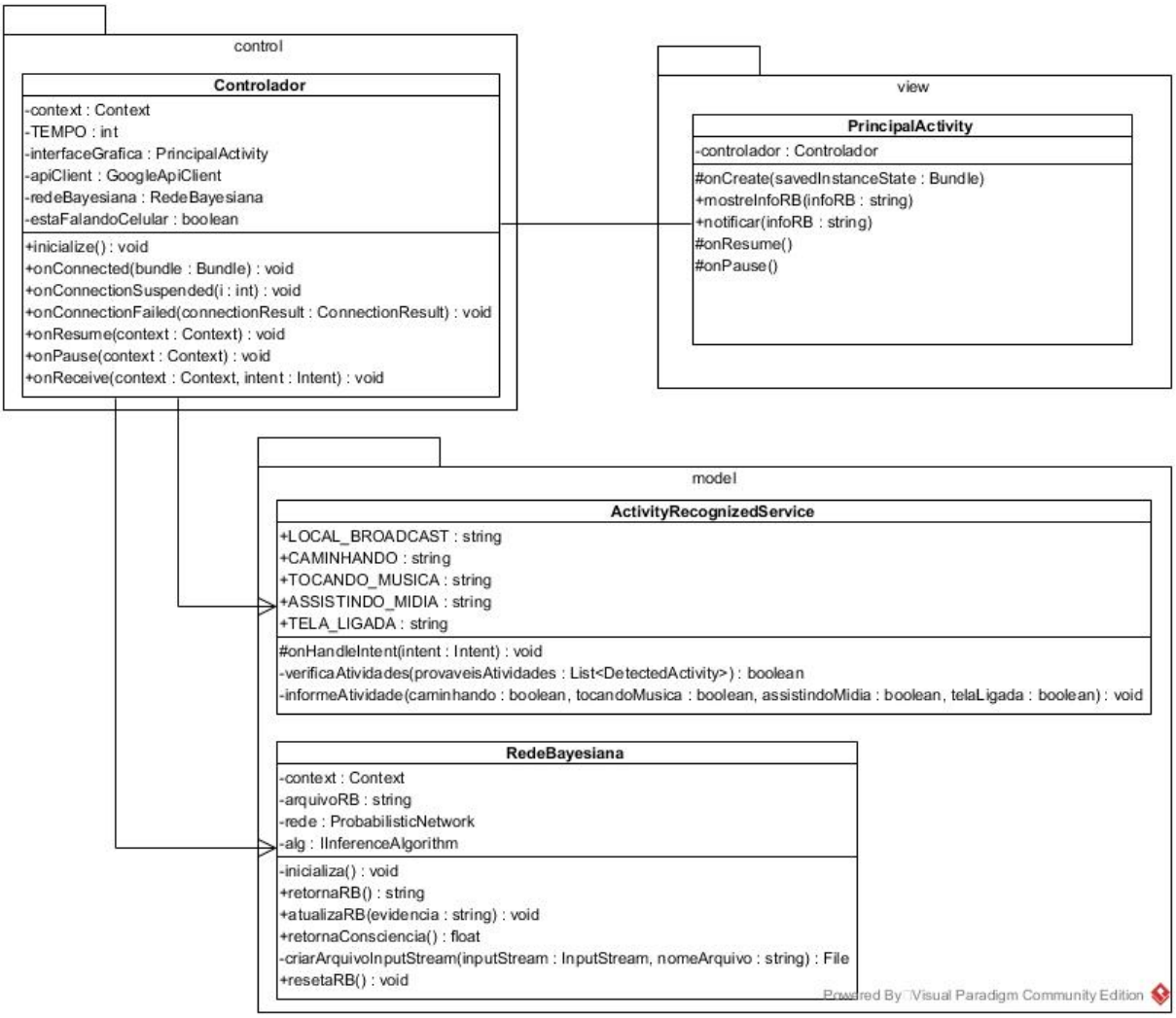


Figura 13 - Diagrama de classes simplificado. Fonte: Do autor (2018)

No pacote view foi criada apenas uma classe, PrincipalActivity, responsável por iniciar a aplicação. Normalmente uma aplicação Android é iniciada por uma classe derivada de uma Activity[9] (ou sua equivalente), sendo a Activity uma classe que representa uma tela com interface do usuário, podendo mostrar informações e tratar toques na tela. Dessa forma, a classe PrincipalActivity também é responsável pela interação com o usuário, e por isso foram criados os métodos mostreInfoRB() e notificar(). Não foi necessário tratar toques na tela. O método mostreInfoRB() imprime na tela as informações relacionadas à RB, e o método notificar() realiza a notificação do usuário abrindo um popup na tela com aviso sonoro, além de um ícone permanecer na barra superior do sistema até que o usuário o visualize.

9"Atividades | Android Developers." https://developer.android.com/guide/components/activities?hl=pt-br. Acessado em 29 mai. 2018.

No pacote control, foi criada a classe Controlador, responsável por fazer a intermediação entre a interface do usuário e o domínio do problema. Uma vez que a aplicação é inicializada pela classe PrincipalActivity, ela imediatamente passa o controle para a classe Controlador. Essa classe fica aguardando que a classe ActivityRecognizedService, ou o próprio sistema operacional no caso de ligação telefônica, informe qualquer um dos usos possíveis do smartphone, conforme definidos neste trabalho, encaminhando os dados obtidos através dos sensores para serem processados na RB. O resultado obtido da inferência bayesiana é então encaminhado para ser mostrado na interface gráfica.

Por fim, no pacote model, foram criadas duas classes: ActivityRecognizedService e RedeBayesiana. Para medir a CS enquanto o usuário está distraído com algum aplicativo do smartphone, foi necessário que a aplicação executasse em background, ou seja, o usuário poderia utilizar qualquer funcionalidade do seu aparelho enquanto a aplicação ficaria monitorando o seu uso. Esse monitoramento é feito utilizando a classe ActivityRecognizedService, que implementa o chamado serviço[10], que fica constantemente verificando os estados dos sensores. No momento que o serviço identifica uma atividade, ele realiza uma transmissão sinalizando para a aplicação que determinada atividade aconteceu. Na Figura 14, adaptada de um tutorial da internet[11], é possível observar esse comportamento, em que o serviço encaminha uma transmissão (sendBroadcast()) para a aplicação, intermediada pelo sistema, que recebe os dados através da classe derivada da Activity. Apenas com relação a identificação de chamada telefônica, ela não é tratada pela classe ActivityRecognizedService, pois o próprio sistema realiza uma transmissão global na ocorrência de ligação, capturada pelo método onReceive() da classe Controlador.

---

10"Serviços | Android Developers." 25 abr. 2018, https://developer.android.com/guide/components/services?hl=pt-br. Acessado em 29 mai. 2018.

11"Android Services - Tutorial - Vogella.com.", http://www.vogella.com/tutorials/AndroidServices/article.html. Acessado em 28 mai. 2018.

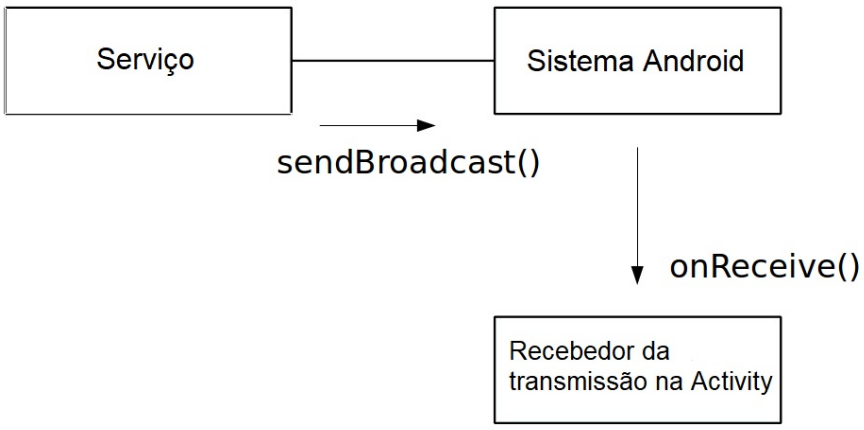


Figura 14 - Funcionamento do serviço. Fonte: Site da Internet (adaptado)

Por fim, a classe RedeBayesiana faz a implementação da RB, fazendo uso da biblioteca do UnbBayes. Através do método atualizaRB() as evidências são recebidas da classe Controlador, que alimentam a RB e que por sua vez propaga as evidências para o restante da rede. O algoritmo de inferência bayesiana utilizado foi o de junção de árvore, por ser o único disponível, uma vez que a utilização de outros algoritmos é feita por meio de plugins, que acabaram tendo que ser retirados devido a incompatibilidade com a API do Android. Foi criado o método retornaRB() para, assim que requisitado, retornar os dados da rede para eventual impressão na tela. Foi necessário ainda criar o método criarArquivoInputStream(), para carregar o arquivo que contém a RB. Optou-se por carregar apenas arquivos .net, uma vez que seria o suficiente para os propósitos deste trabalho, devendo o arquivo estar dentro da pasta assets do projeto.

### 4.4.4 Saída do Sistema

A saída do sistema é composta pelos alertas emitidos ao usuário e por informações que são impressas na interface gráfica, que ocorrem após o processamento interno dos dados. O alerta pode ser uma mensagem de texto que aparece na parte inferior da tela por cerca de 3 segundos, ou pode vir na forma de uma notificação na barra de status que fica na parte superior da tela, que se apresenta juntamente com um popup de mensagem e um aviso sonoro, além de fazer vibrar o aparelho.

Os avisos são mostrados em dois momentos: (1) quando é identificada a caminhada juntamente com alguma distração; e (2) quando o nível de consciência é

considerado baixo. Na primeira situação, é impressa na interface gráfica informações relacionadas à RB e qual tipo de distração ocorreu. Além disso, ocorre a notificação do usuário por meio da mensagem "Atenção! Caminhar e utilizar o smartphone não é seguro!". No segundo momento, é avaliada a consciência. Não havendo evidência inserida na rede, ela apresenta a probabilidade de 80% para Consciente. Por outro lado, havendo a inserção de evidência no nodo DistracaoApp para a ocorrência de distração, a probabilidade cai para 77,1% para Consciente, como é possível observar da Figura 15. Dessa forma, o alerta ao usuário acontece quando o resultado da RB retornar um valor menor que 79%, feito por meio da mensagem "Nível de consciência abaixo de 0.79!", e uma notificação com o aviso "Atenção! Você pode estar distraído!".

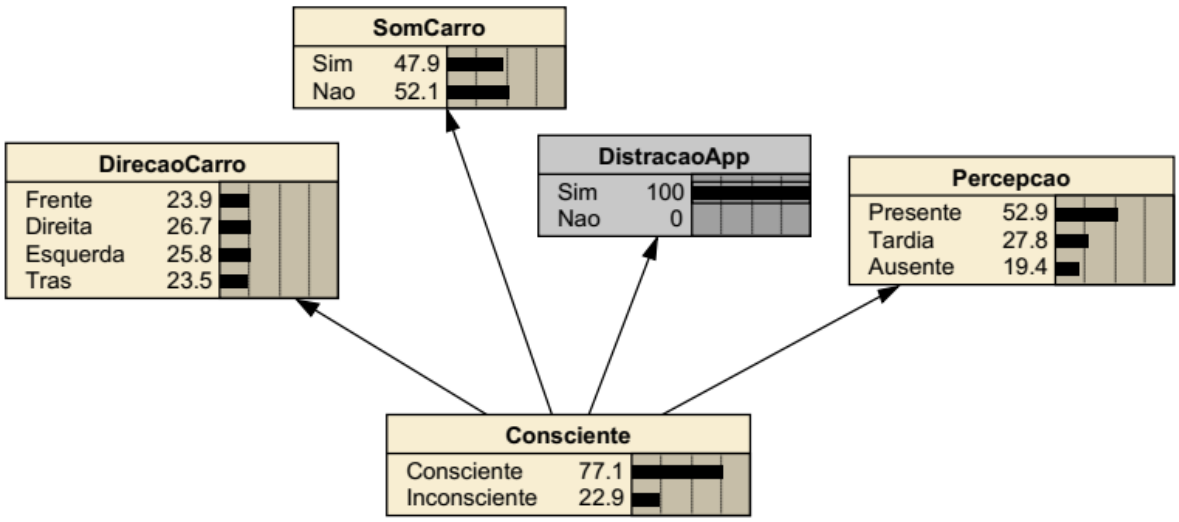


Figura 15 - RB com evidência no nodo DistacaoApp. Fonte: Do autor (2018)

## 4.5 TESTES

Os testes foram realizados apenas com relação ao funcionamento do aplicativo, ou seja, se o aplicativo conseguia identificar o uso do smartphone com base em dados dos sensores e realizar o processamento adequado, mostrando no final informações ao usuário. Não foram realizados testes em ambientes urbanos, como estradas e ferrovias, tampouco para medir a eficácia do modelo em inferir a CS.

Para os testes foi utilizado um smartphone da marca Motorola, modelo Moto C Plus, rodando o Android na versão 7.0, equipado apenas com GPS e

acelerômetro, não possuindo bússola, giroscópio e sensor de proximidade. A área de um corredor com cerca de 9 metros de comprimento foi suficiente para os testes. Para visualizar as capturas de tela dos testes vide Apêndice A.

Os primeiros testes ocorreram com a biblioteca do UnbBayes dentro do aplicativo. Desconsiderando os testes que apontaram a incompatibilidade, os testes iniciais foram satisfatórios, com o motor de inferência bayesiana apresentando os resultados de forma imediata e praticamente idênticos aos obtidos com o software Hugin, com uma pequena diferença no arredondamento das casas decimais.

Foram realizados testes preliminares de cada sensor, analisando-os separadamente:

- Uma vez inicializada a aplicação, a identificação da caminhada teve tempos de resposta entre 2 e 30 segundos. Não foi possível identificar o motivo de isso acontecer, provavelmente deve estar ligado à prioridade que o sistema concede ao serviços que rodam em background. Também pode estar relacionado com a maneira que a API caracteriza a caminhada, uma vez que pode-se caminhar rápida ou lentamente, segurando o dispositivo firme ou de forma relaxada;
- A identificação de tela ligada aconteceu de modo satisfatório, com resposta imediata;
- A identificação de sons tocando, uma vez inicializada a aplicação, ocorreu com tempos de resposta entre 3 e 15 segundos, provavelmente também devido à prioridade concedida pelo sistema. O aplicativo consegue identificar sons gerados tanto por um tocador de música como por um player de vídeo. Testes com a visualização de vídeos na internet foram positivos;
- A identificação de chamada telefônica ocorreu de modo satisfatório, tanto para chamadas realizadas como para recebidas, com resposta imediata.

Os testes com a versão final do aplicativo mostraram os mesmos comportamentos dos sensores, mas necessariamente foram realizados utilizando o smartphone e caminhando ao mesmo tempo, o que significa que todos os testes

finais tiveram tempos de resposta entre 3 e 30 segundos. Nos testes a tela também deveria estar ligada, com a exceção de escutar música.

Foram também realizados testes com as notificações ao usuário. As notificações se mostraram adequadas pois conseguiam se sobrepor a qualquer aplicação em uso. Por exemplo, ao caminhar e assistir a um vídeo na internet, o aplicativo exibe uma mensagem na tela por cerca de 3 segundos, sobrepondo o vídeo em execução, além de um aviso sonoro que também se sobrepõe ao som gerado pelo vídeo, fazendo ainda o aparelho vibrar e mostrando um ícone na barra superior da tela.

### 4.5.1 Limitações

Durante os testes foram identificadas algumas limitações:

- Para conseguir analisar o uso do dispositivo móvel, o aplicativo precisa ser executado pelo menos uma vez, após a instalação ou depois de ligar ou reiniciar o aparelho;
- Depois que o aplicativo é iniciado, se deixar de usá-lo, ele para de monitorar o uso do dispositivo após algumas horas. Não foi possível fazer a análise dessa limitação a fundo, mas é provável que a aplicação principal deixa de executar pela inatividade, e dessa forma não processando os dados recebidos pelos serviços. Outra possível causa é o serviço deixar de executar em background, provavelmente parado pelo sistema operacional, devido a inatividade;
- Os testes com som apresentaram comportamento indesejado, em que o aplicativo ficava repetindo a notificação por mais algumas vezes mesmo depois de desligar o som.

# 5 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho se propôs a desenvolver um aplicativo que fizesse uso de RB, o que foi alcançado com a utilização da API do UnbBayes. Era essencial que a inserção de evidências e inferência bayesiana fosse processada dentro da aplicação em tempo real, pois em uma situação extrema a resposta precisa ser imediata, considerando que o que está em jogo é a segurança do usuário. O processamento da inferência bayesiana em um servidor remoto nesse caso seria inviável, pois poderia ocorrer uma falha na comunicação ou mesmo uma demora na resposta. O uso do UnbBayes se mostrou adequado, sendo esta uma ferramenta open source, mostrando alta compatibilidade com o sistema Android.

Uma vez que ainda são poucas as ferramentas que permitem a utilização de IA em aplicativos móveis, a adaptação da API do UnbBayes possibilitando o uso de RB em uma aplicação Android pode ser considerada uma contribuição deste trabalho para o desenvolvimento de aplicativos que trazem a IA para dentro do dispositivo.

Este trabalho também propôs que o aplicativo fosse desenvolvido para a segurança dos pedestres, o que foi alcançado em parte, tendo em vista as limitações impostas pela plataforma e pelo modelo. Em uma situação ideal a implementação de um mecanismo de segurança no smartphone deveria ser feito pela proprietária do sistema operacional, considerando a necessidade da aplicação rodar em background e a dependência de acesso a hardware. Mas ainda assim, o aplicativo consegue analisar de modo satisfatório o uso do aparelho por meio de dados dos sensores, processando esses dados e chegando à conclusões com relação ao nível de CS, notificando o usuário por meio de avisos visuais e sonoros.

Apesar do modelo de CS não considerar a caminhada, foi percebido durante os testes que faria mais sentido que o aplicativo fizesse a notificação do usuário somente na ocorrência de caminhada, pois seria a situação real de risco, pois do contrário o usuário seria incomodado com as notificações mesmo nas situações normais de uso do aparelho.

Uma das motivações do trabalho foi contribuir com o projeto Awareness, e acredita-se que isso tenha sido alcançado com a construção do aplicativo e do uso do modelo de consciência situacional que foi desenvolvido no projeto. É importante

destacar que houveram algumas limitações do uso do modelo, pois percebe-se que o mesmo foi construído com base na problemática estudada, que levou em conta questões como a oclusão e a percepção do ponto de vista do usuário, mas questões essas que seriam difíceis de se detectar somente com os sensores. Talvez em um futuro próximo seja possível identificar a direção de qualquer veículo em relação ao dispositivo móvel.

Dessa forma, conclui-se que todos os objetivos foram alcançados.

## 5.1 TRABALHOS FUTUROS

A partir do desenvolvimento aqui realizado, foram identificadas possíveis oportunidades de trabalhos futuros.

Seria possível, por exemplo, melhorar o projeto UnbBayes para Android, deixando mais organizado. Seria interessante que o projeto fosse documentado, algo que não foi possível durante o desenvolvimento deste trabalho, o que facilitaria bastante a sua melhoria. Algo que traria grandes melhorias no UnbBayes para Android seria possibilitar o uso dos plugins, função que foi retirada por causa da incompatibilidade, o que permitiria explorar outras funcionalidades, como o uso de ontologias.

Uma possibilidade seria melhorar o aplicativo aqui desenvolvido. Poderiam ser estudadas maneiras de se evitar que o monitoramento do uso do smartphone fosse interrompido, ou implementar a identificação de sons de veículos para possibilitar a inserção de evidência no nodo SomCarro, o que não se conseguiu desenvolver neste trabalho.

Outra sugestão seria utilizar a aplicação Android dentro do projeto Awareness, validando a aplicação dentro do ambiente simulado, considerando a complexidade de se testar dentro de um ambiente real, devido aos riscos inerentes. Haveria a necessidade de se ampliar o projeto Awareness considerando a caminhada, pois é uma condição do aplicativo para indicar um nível de consciência baixo.

Algo que poderia ser realizado dentro do projeto Awareness é o estudo de diferentes tipos de distrações causadas pelo uso do smartphone, como ouvir música, falar no celular, assistir vídeo, jogar e encaminhar mensagens, e se elas isoladamente têm maior impacto na CS, pois não importava qual era a distração, todas eram classificadas como evidências a serem inseridas no nodo DistracaoApp.

Da mesma forma, poderia-se construir uma rede bayesiana considerando outras possibilidades de se inserir evidências, pois neste trabalho a inserção de evidência ficou restrita a um único nodo. A identificação de caminhada e de tela ligada, por exemplo, poderiam ser nodos na rede. Para isso seria necessário verificar a disponibilidade de dados para se calcular a distribuição de probabilidades, ou realizar novos experimentos dentro do ambiente simulado.

# 6. REFERÊNCIAS

ARA-SOUZA, Anderson Luiz. **REDES BAYESIANAS: UMA INTRODUÇÃO APLICADA A CREDIT SCORING.** 2010. 99 f. TCC (Graduação) - Curso de Estatística, Universidade Federal de São Carlos – Ufscar, São Carlos, 2010.

BUCZAK, Anna L.; GUVEN, Erhan. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. **Ieee Communications Surveys & Tutorials.** p. 1153-1176. julho-dezembro. 2016.

CHARNIAK, Eugene. Bayesian Networks without Tears. **Ai Magazine.** Menlo Park, (USA), p. 50-63. winter 1991. Disponível em: <https://www.aaai.org/ojs/index.php/aimagazine/article/view/918/836>. Acesso em: 28 jun. 2017.

ENDSLEY, Mica R.. Toward a Theory of Situation Awareness in Dynamic Systems. **Human Factors: The Journal of the Human Factors and Ergonomics Society,** [s.l.], v. 37, n. 1, p.32-64, 1 mar. 1995. SAGE Publications. http://dx.doi.org/10.1518/001872095779049543.

LEE, Young-seol; CHO, Sung-bae. An Efficient Energy Management System for Android Phone Using Bayesian Networks. **2012 32nd International Conference On Distributed Computing Systems Workshops,** Macau, China, p.102-107, jun. 2012. IEEE. http://dx.doi.org/10.1109/icdcsw.2012.47.

LOBATO, Antonio Gonzalez Pastana; ANDREONI LOPEZ, M.; DUARTE, O. C. M. B. Um sistema acurado de detecção de ameaças em tempo real por processamento de fluxos. **XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC**, 2016.

LUGER, George F. 2014. **Inteligência artificial.** 6. ed. São Paulo: Pearson Education do Brasil, c2014. xvii, 614 p. ISBN 9788581435503.

MATSUMOTO, Shou et al. **UnBBayes: a java framework for probabilistic models in AI.** Java in academia and research, p. 34, 2011.

NEAPOLITAN, R. E. **Learning Bayesian Networks.** Upper Saddle River: Pearson, 2004.

PEREIRA, Rodolfo Lottin. **Aplicação de Aprendizagem por Reforço para um Modelo Bayesiano de Consciência Situacional.** 2017. 119f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade Federal de Santa Catarina – UFSC, Florianópolis, 2017.

SANTOS, Elder Rizzon; FAZENDA, B. M.. **Computational Model of Situational Awareness for users of smartphones in the vicinity of traffic.** [S.l.], 2016.

SARABANDO, Ana Cristina Lopes. **Um estudo do comportamento de Redes Bayesianas no prognóstico da sobrevivência no cancro da próstata.** 2010. 84 f. Dissertação (Mestrado) - Curso de Medicina, Universidade do Porto, Porto, Portugal, 2010. Disponível em: <https://repositorio-aberto.up.pt/bitstream/10216/55452/2/tesedo MIMVF.pdf>. Acesso em: 28 jun. 2017.

YI, Si-hyuk; CHO, Sung-bae. A Battery-Aware Energy-Efficient Android Phone with Bayesian Networks. **2012 9th International Conference On Ubiquitous Intelligence And Computing And 9th International Conference On Autonomic And Trusted Computing,** Fukuoka, Japan, p.204-209, set. 2012. IEEE. http://dx.doi.org/10.1109/uic-atc.2012.157.

SCHWEBEL, David C. et al. Distraction and pedestrian safety: How talking on the phone, texting, and listening to music impact crossing the street. **Accident Analysis & Prevention**, Birmingham, USA, v. 45, p.266-271, mar. 2012. Elsevier BV. http://dx.doi.org/10.1016/j.aap.2011.07.011.

# APÊNDICE A - Capturas de Tela dos Testes

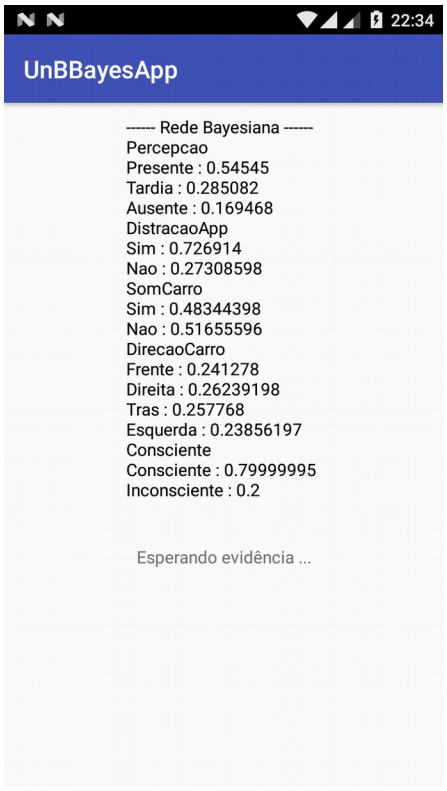


Figura A-1 - Tela inicial

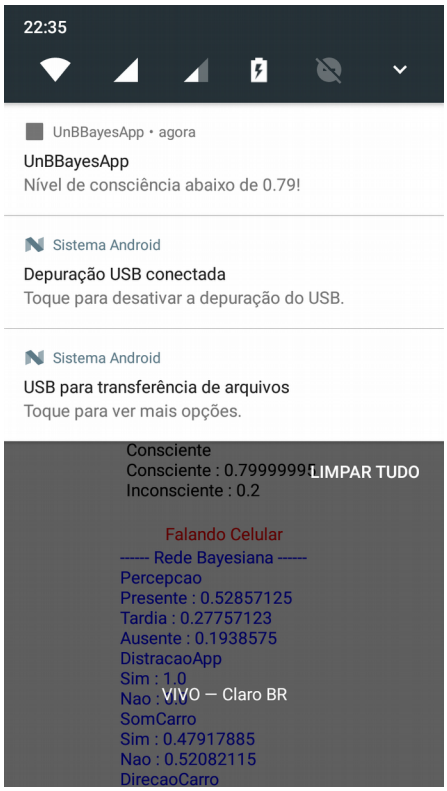


Figura A-2 - Exemplo de notificação

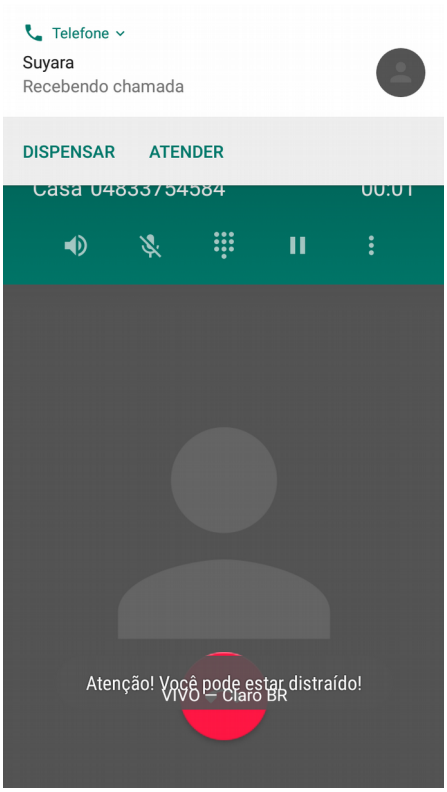


Figura A-3 - Chamada e aviso

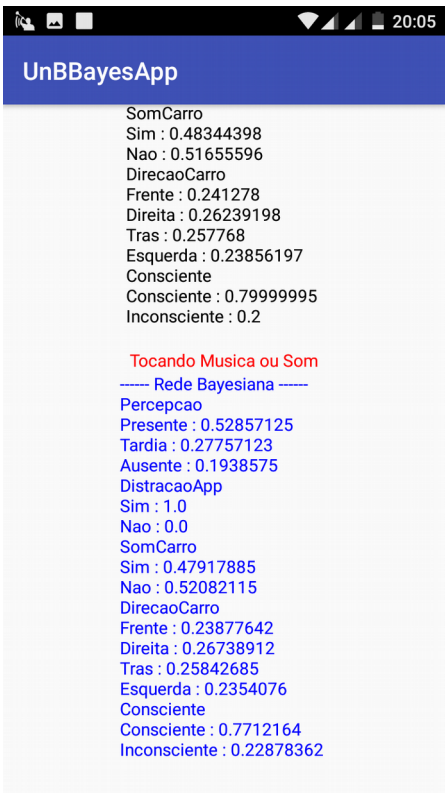


Figura A-4 - Resultado da inferência 77,12%

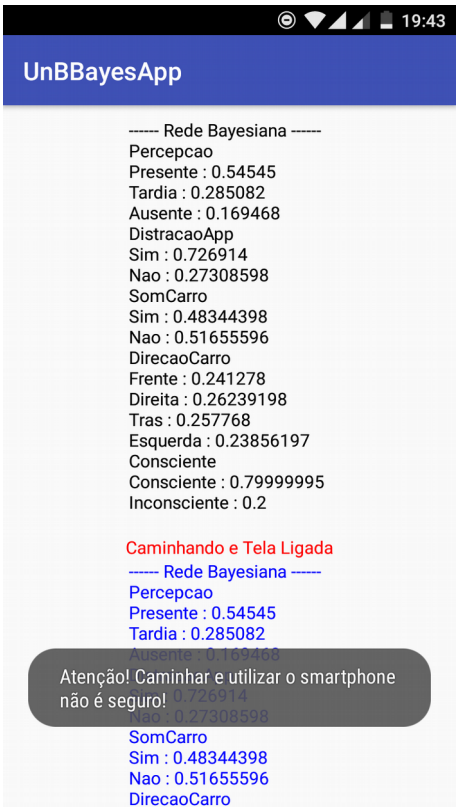


Figura A-5 - Aviso de caminhada

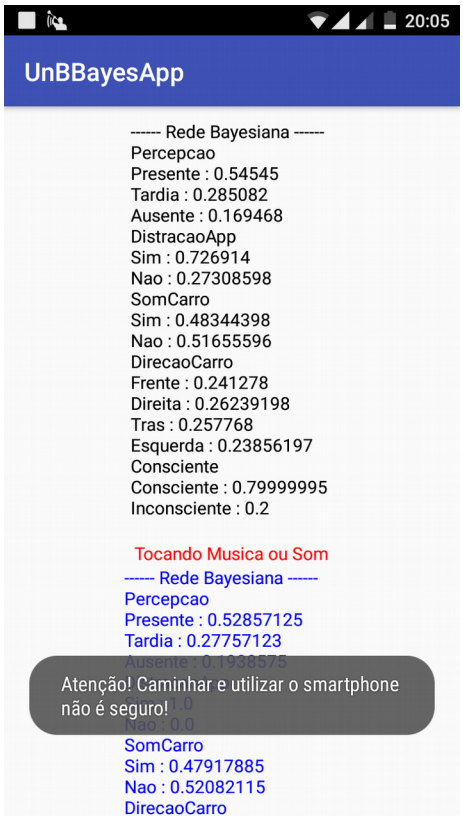


Figura A-6 - Caminhando e ouvindo música

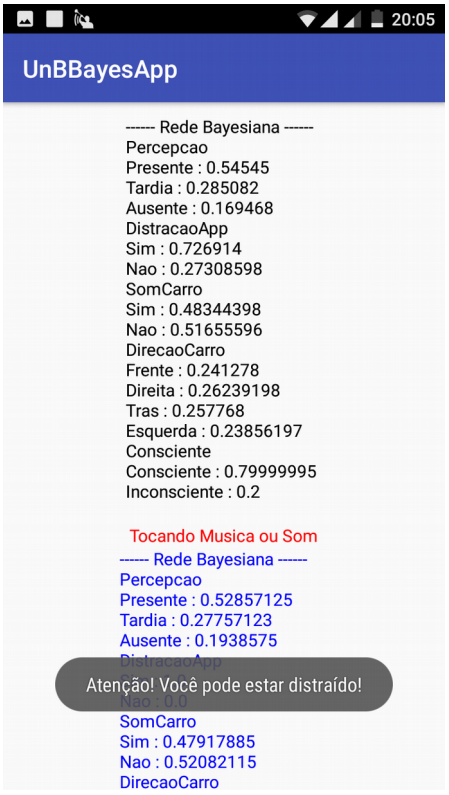


Figura A-7 - Aviso de distração

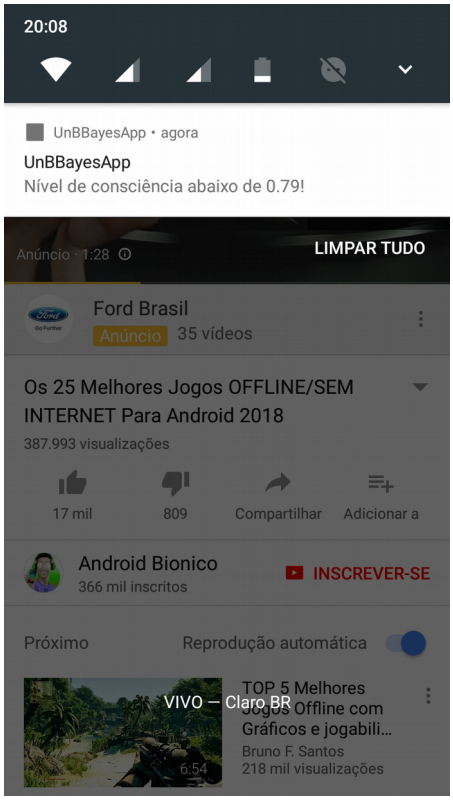


Figura A-8 - Notificação e assistindo vídeo


Figura A-9 - Aviso de caminhada e assistindo vídeo

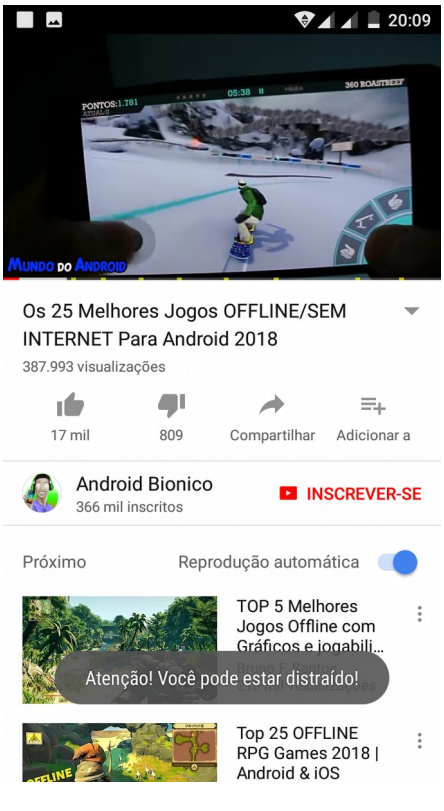


Figura A-10 - Aviso de distração e assistindo vídeo

# APÊNDICE B - Código

O código do aplicativo desenvolvido na plataforma Android foi juntado abaixo, mas é possível fazer o download do projeto completo de um repositório na internet, pois há outros arquivos que são necessários para executar o aplicativo.

O código do projeto Android e um arquivo instalador .apk estão disponíveis em https://github.com/edumk/unbbayesapp. O arquivo .apk deve ser executado de dentro do smartphone para ser instalado.

O código do UnbBayes adaptado para Android está disponível em https://github.com/edumk/unbbayesandroid. Foram juntadas as últimas três versões, sendo a 4.0 a mais estável. Também se encontra no repositório o arquivo .jar, o qual pode ser utilizado como biblioteca em outros projetos.

```java
package com.teste.eduardo.unbayesteste.view;

import android.app.Notification;
import android.app.NotificationManager;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.NotificationCompat;
import android.support.v4.app.NotificationManagerCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;
import android.widget.Toast;

import com.teste.eduardo.unbayesteste.R;
import com.teste.eduardo.unbayesteste.control.Controlador;

/**
 * Classe que implementa a interface grafica.
 * Mostra informacoes sobre a RB na interface grafica do aplicativo.
 * Tambem notifica o usuario sobre o nivel de consciencia.
 * Tambem exibe mensagens de avisos na tela (Toast)
 */

public class PrincipalActivity extends AppCompatActivity {

        private Controlador controlador;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_principal);

        //Solicita permissao para gerenciar ligacoess em tempo de execucao
        if (ContextCompat.checkSelfPermission(this,
        android.Manifest.permission.READ_PHONE_STATE)
                != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,new
            String[]{android.Manifest.permission.READ_PHONE_STATE},1);
        }

        controlador = new Controlador();

        //Registra um filtro procurando por ligacao
        IntentFilter intentFilter = new
        IntentFilter("android.intent.action.PHONE_STATE");
        registerReceiver(controlador,intentFilter);

        controlador.inicialize(this);
    }

    public void mostreInfoRB(String infoRB) {
        TextView texto = findViewById(R.id.rb);
        texto.setTextColor(Color.BLACK);
        texto.setText(infoRB);
    }

    public void mostreInfoEvidencia(String infoRB) {
        TextView texto = findViewById(R.id.evidencia);
        texto.setTextColor(Color.RED);
        texto.setText(infoRB);
    }

    public void mostreInfoInferencia(String infoRB) {
        TextView texto = findViewById(R.id.inferencia);
```

```
67          texto.setTextColor(Color.BLUE);
68          texto.setText(infoRB);
69      }
70
71      //Notifica com icone, som e vibracao
72      public void notificar(String infoRB) {
73          NotificationCompat.Builder notificacao = new NotificationCompat.Builder(this);
74          notificacao.setContentText(infoRB);
75          notificacao.setSmallIcon( R.mipmap.ic_launcher );
76          notificacao.setContentTitle( getString( R.string.app_name ) );
77          Uri soundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
78          notificacao.setSound(soundUri);
79          long[] v = {500,1000};
80          notificacao.setVibrate(v);
81          NotificationManagerCompat.from(this).notify(0, notificacao.build());
82      }
83
84      public void notificarPopup(String infoRB) {
85          Toast.makeText(this, infoRB, Toast.LENGTH_LONG).show();
86      }
87
88      @Override
89      protected void onResume() {
90          super.onResume();
91          controlador.onResume(this);
92      }
93
94      @Override
95      protected void onPause() {
96          super.onPause();
97      }
98
99  }
100
```

PrincipalActivity.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.teste.eduardo.unbayesteste.view.PrincipalActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:padding="10dp"
            android:layout_centerHorizontal="true">

            <TextView
                android:id="@+id/rb"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerHorizontal="true"
                android:text="Hello World!" />

            <TextView
                android:id="@+id/evidencia"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_below="@id/rb"
                android:layout_centerHorizontal="true"
                android:text="" />

            <TextView
                android:id="@+id/inferencia"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_below="@id/evidencia"
                android:layout_centerHorizontal="true"
                android:text="Esperando evidência ..." />

        </RelativeLayout>
    </ScrollView>

</android.support.constraint.ConstraintLayout>
```

activity_principal.xml

```java
package com.teste.eduardo.unbayesteste.control;

import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.content.LocalBroadcastManager;
import android.telephony.TelephonyManager;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.ActivityRecognition;
import com.teste.eduardo.unbayesteste.model.ActivityRecognizedService;
import com.teste.eduardo.unbayesteste.model.RedeBayesiana;
import com.teste.eduardo.unbayesteste.view.PrincipalActivity;

/**
 * Created by Eduardo on 12/12/2017.
 * Classe que faz a intermediacao entre a interface grafica e o dominio do problema.
 * Faz alguns tratamentos das informacoes que chegam.
 */

public class Controlador extends BroadcastReceiver implements
        GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener {

    private Context context;
    private int TEMPO = 3000; // segundos
    private PrincipalActivity interfaceGrafica;
    private GoogleApiClient apiClient;
    private RedeBayesiana redeBayesiana;

    private boolean estaFalandoCelular;

    public Controlador( ) {
    }

    public void inicialize(PrincipalActivity principalActivity) {
        this.context = principalActivity.getBaseContext();
        interfaceGrafica = principalActivity;
        redeBayesiana = new RedeBayesiana(context);
        estaFalandoCelular = false;

        //Inicializa a API para identificar caminhada
        apiClient = new GoogleApiClient.Builder(context)
                .addApi(ActivityRecognition.API)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .build();

        apiClient.connect();

        String infoRB = null;
        try {
            infoRB = redeBayesiana.retornaRB();
        } catch (Exception e) {
            e.printStackTrace();
        }
        interfaceGrafica.mostreInfoRB(infoRB);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        //Cria um intent, ou token, que permite o servico tratar a identificacao de
        caminhada em um tempo futuro
        Intent intent = new Intent(context, ActivityRecognizedService.class);
```

```
69          PendingIntent pendingIntent = PendingIntent.getService( context, 0, intent,
            PendingIntent.FLAG_UPDATE_CURRENT );
70          ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates(apiClient,
            TEMPO, pendingIntent);
71      }
72
73      @Override
74      public void onConnectionSuspended(int i) {
75      }
76
77      @Override
78      public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {}
79
80      public void onResume(PrincipalActivity principalActivity) {
81          //Registra um receiver localmente
82          LocalBroadcastManager.getInstance(context)
83                  .registerReceiver(this, new
                    IntentFilter(ActivityRecognizedService.LOCAL_BROADCAST));
84      }
85
86      public void onPause(PrincipalActivity principalActivity) {
87      }
88
89      //Recebe informacoes dos servicos que rodam em background
90      @Override
91      public void onReceive(Context context, Intent intent) {
92          boolean estaCaminhando = false;
93          boolean telaLigada = false;
94          boolean estaTocandoMusica = false;
95
96          //Trata as informacoes relacionadas aos sensores
97          if(intent.getAction().equals(ActivityRecognizedService.LOCAL_BROADCAST)){
98              estaCaminhando =
                intent.getBooleanExtra(ActivityRecognizedService.CAMINHANDO, false);
99              telaLigada = intent.getBooleanExtra(ActivityRecognizedService.TELA LIGADA,
                false);
100             estaTocandoMusica =
                intent.getBooleanExtra(ActivityRecognizedService.TOCANDO_MUSICA, false);
101         }
102         //Trata as informacoes relacionadas a ligacoes
103         if(intent.getAction().equals("android.intent.action.PHONE_STATE")) {
104             String stateStr = intent.getExtras().getString(TelephonyManager.EXTRA STATE);
105             if(stateStr.equals(TelephonyManager.EXTRA_STATE_OFFHOOK)) {
106                 estaFalandoCelular = true;
107             }
108         }
109         if(estaCaminhando) {
110             interfaceGrafica.mostreInfoEvidencia("Caminhando");
111             if(telaLigada){
112                 interfaceGrafica.mostreInfoEvidencia("Tela Ligada");
113                 interfaceGrafica.notificarPopup("Atenção! Caminhar e utilizar o
                    smartphone não é seguro!");
114                 if(estaTocandoMusica) {
115                     interfaceGrafica.mostreInfoEvidencia("Tocando Musica ou Som");
116                     redeBayesiana.atualizaRB("DistracaoApp");
117                 }
118                 if(estaFalandoCelular){
119                     interfaceGrafica.mostreInfoEvidencia("Falando Celular");
120                     redeBayesiana.atualizaRB("DistracaoApp");
121                 }
122             } else {
123                 if(estaTocandoMusica) {
124                     interfaceGrafica.notificar("Atenção! Caminhar e utilizar o
                        smartphone não é seguro!");
125                     interfaceGrafica.mostreInfoEvidencia("Tocando Musica ou Som");
126                     redeBayesiana.atualizaRB("DistracaoApp");
127                 }
128             }
129             interfaceGrafica.mostreInfoInferencia(redeBayesiana.retornaRB());
```

```
130             }
131
132             //Depois de tratadas as informacoes, mede o nivel de consciencia e notifica
133             Float consciencia = redeBayesiana.retornaConsciencia();
134             if(consciencia < 0.79){
135                 interfaceGrafica.notificar("Nível de consciência abaixo de 0.79!");
136                 interfaceGrafica.notificarPopup("Atenção! Você pode estar distraído!");
137                 estaFalandoCelular = false;
138                 redeBayesiana.resetaRB();
139             }
140         }
141     }
142
```

Controlador.java

```java
package com.teste.eduardo.unbayesteste.model;

import android.app.IntentService;
import android.content.Context;
import android.content.Intent;
import android.media.AudioManager;
import android.os.PowerManager;
import android.support.v4.content.LocalBroadcastManager;

import com.google.android.gms.location.ActivityRecognitionResult;
import com.google.android.gms.location.DetectedActivity;

import java.util.List;

/**
 * Created by Eduardo on 29/08/2017.
 * Classe responsavel por identificar atividades nos sensores.
 * Implementa o servico que roda em background.
 */

public class ActivityRecognizedService extends IntentService {

    public static String LOCAL_BROADCAST = "LOCAL_BROADCAST";
    public static String CAMINHANDO = "CAMINHANDO";
    public static String TOCANDO_MUSICA = "TOCANDO_MUSICA";
    public static String ASSISTINDO_MIDIA = "ASSISTINDO_MIDIA";
    public static String TELA_LIGADA = "TELA_LIGADA";

    public ActivityRecognizedService() {
        super("ActivityRecognizedService");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        boolean estaCaminhando = false;
        boolean estaTocandoMusica = false;
        boolean telaLigada = false;

        if(ActivityRecognitionResult.hasResult(intent)) {
            ActivityRecognitionResult resultado =
            ActivityRecognitionResult.extractResult(intent);
            estaCaminhando = verificaAtividades( resultado.getProbableActivities() );
        }
        AudioManager manager = (AudioManager)getSystemService(Context.AUDIO_SERVICE);
        if(manager.isMusicActive()) {
            estaTocandoMusica = true;
        }

        // Usada em api menor que API20?
        PowerManager powerManager = (PowerManager) getSystemService(POWER_SERVICE);
        if (powerManager.isScreenOn()){
            telaLigada = true;
        }

        informeAtividade(estaCaminhando, estaTocandoMusica, telaLigada);
    }

    private boolean verificaAtividades(List<DetectedActivity> provaveisAtividades) {
        boolean estaCaminhando = false;
        for( DetectedActivity atividade : provaveisAtividades ) {
            switch( atividade.getType() ) {
                case DetectedActivity.ON_FOOT: {
                    if( atividade.getConfidence() >= 75 ) {
                        estaCaminhando = true;
                    }
                    break;
                }
                case DetectedActivity.RUNNING: {
                    if( atividade.getConfidence() >= 75 ) {
```

```
69                          estaCaminhando = true;
70                      }
71                      break;
72                  }
73                  case DetectedActivity.WALKING: {
74                      if( atividade.getConfidence() >= 75 ) {
75                          estaCaminhando = true;
76                      }
77                      break;
78                  }
79              }
80          }
81          return estaCaminhando;
82      }
83
84      private void informeAtividade(boolean caminhando, boolean tocandoMusica, boolean
        telaLigada) {
85          Intent intent = new Intent(LOCAL_BROADCAST);
86          intent.putExtra(CAMINHANDO, caminhando);
87          intent.putExtra(TOCANDO_MUSICA, tocandoMusica);
88          intent.putExtra(TELA LIGADA, telaLigada);
89          LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
90      }
91  }
92
```

ActivityRecognizedService.java

```java
package com.teste.eduardo.unbayesteste.model;

import android.content.Context;
import android.content.res.AssetManager;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import unbbayes.io.NetIO;
import unbbayes.prs.Node;
import unbbayes.prs.bn.JunctionTreeAlgorithm;
import unbbayes.prs.bn.ProbabilisticNetwork;
import unbbayes.prs.bn.ProbabilisticNode;
import unbbayes.util.extension.bn.inference.IInferenceAlgorithm;

/**
 * Created by Eduardo on 12/12/2017.
 * Classe responsavel por carregar a RB e realizar a inferencia.
 * Faz uso da biblioteca do UnbBayes.
 */

public class RedeBayesiana {

    private Context context;
    private String arquivoRB;
    private ProbabilisticNetwork rede;
    private IInferenceAlgorithm alg;

    public RedeBayesiana(Context context) {
        this.context = context;
        AssetManager am = context.getAssets();
        InputStream inputStream = null;

        //Nome do arquivo a ser carregado
        arquivoRB = "ConscienciaSituacional.net";
        try {
            inputStream = am.open(arquivoRB);
        } catch (IOException e) {
            e.printStackTrace();
        }
        criarArquivoInputStream(inputStream, arquivoRB);
        try {
            inicializa();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void inicializa() throws Exception {
        rede = (ProbabilisticNetwork) new NetIO().load(new File(context.getFilesDir(),
        arquivoRB));
        // prepara o algoritmo para compilar a rede
        alg = new JunctionTreeAlgorithm();
        alg.setNetwork(rede);
        alg.run();
    }

    public String retornaRB() {
        String redeImpressa = "------ Rede Bayesiana ------"+
        System.getProperty("line.separator");
        for (Node node : rede.getNodes()) {
            redeImpressa = redeImpressa + node.getName() +
            System.getProperty("line.separator");
            for (int i = 0; i < node.getStatesSize(); i++) {
                redeImpressa = redeImpressa + node.getStateAt(i) + " : "
                        + ((ProbabilisticNode)node).getMarginalAt(i) +
```

```
                        System.getProperty("line.separator");
                }
            }
            return redeImpressa;
        }

        public void atualizaRB(String evidencia) {
            // insere evidencia (finding) na rede considerando o no
            ProbabilisticNode noEvidencia = (ProbabilisticNode) rede.getNode(evidencia);
            noEvidencia.addFinding(0); // o estado eh agora 100%
            // propaga a evidencia
            alg.propagate();
        }

        public Float retornaConsciencia() {
            Node node = rede.getNode("Consciente");
            Float valorConsciencia = ((ProbabilisticNode)node).getMarginalAt(0);
            return valorConsciencia;
        }

        public void resetaRB(){
            alg.reset();
        }

        //Carrega o arquivo localizado na pasta assets
        private File criarArquivoInputStream(InputStream inputStream, String nomeArquivo) {
            try {
                File arquivo = new File(context.getFilesDir(), nomeArquivo);
                OutputStream outputStream = new FileOutputStream(arquivo);
                byte buffer[] = new byte[1024];
                int length = 0;
                while ((length = inputStream.read(buffer)) > 0) {
                    outputStream.write(buffer, 0, length);
                }
                outputStream.close();
                inputStream.close();
                return arquivo;
            } catch (IOException e) {
                e.printStackTrace();
            }
            return null;
        }
    }

```

RedeBayesiana.java

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.teste.eduardo.unbayesteste">

    <uses-permission
    android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
    <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".view.PrincipalActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".model.ActivityRecognizedService" />
    </application>

</manifest>
```

AndroidManifest.xml

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.teste.eduardo.unbayesteste"
        minSdkVersion 21
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        multiDexEnabled true
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
            'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.google.android.gms:play-services:11.6.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
    implementation files('libs/unbbayesv2.jar')
}

```

build.gradle

# APÊNDICE C - Artigo

# Aplicação de Abordagens Probabilísticas em Dispositivos Móveis para a Segurança de Pedestres

**Eduardo Massao Kobayashi**

Departamento de Informática e Estatística - Universidade Federal de Santa Catarina (UFSC)

Campus Universitário – Florianópolis – SC - Brasil

eduardokobayashi@live.com

***Abstract.*** *Studies point to the increase of accidents in urban environments due to the intense use of mobile devices, which take the attention of the user. As part of a larger project that seeks to develop a computational model to measure the level of user perception, this paper is intended to develop a mobile application that uses a Bayesian network to represent situational awareness. The application was developed in the Android platform, assuming that the Bayesian inference algorithm was executed within the application, using the UnbBayes library. Tests were performed with the application and the result is presented.*

***Resumo.*** *Estudos apontam para o aumento de acidentes em ambientes urbanos devido ao uso intenso de dispositivos móveis, que tiram a atenção do usuário. Como parte de um projeto maior, que busca desenvolver um modelo computacional que possibilite medir o nível de percepção do usuário, este trabalho se destina a desenvolver um aplicativo para dispositivo móvel que utilize uma rede bayesiana para representar a consciência situacional. O aplicativo foi desenvolvido na plataforma Android, tendo como premissa que o algoritmo de inferência bayesiana fosse executado dentro da aplicação, utilizando-se para isso a biblioteca do UnbBayes. Foram realizados testes com a aplicação e o resultado é apresentado.*

## 1. Introdução

O uso de dispositivos móveis tem aumentado gradativamente, sendo um fenômeno global. Mas seu uso perto de estradas e ferrovias tem sido motivo de preocupação devido ao também crescente número de acidentes e fatalidades que vêm acontecendo devido ao uso desatento do dispositivo. O que se percebe é que o uso intenso do aparelho causa a distração cognitiva do usuário, que não percebe as eventuais situações de risco no local em que se encontra. A partir disso, surgiu um projeto que se propõe a desenvolver um modelo computacional de consciência situacional para usuários de dispositivos móveis (smartphones, tablets, etc.) quando perto de estradas e ferrovias, intitulado Road Awareness (SANTOS; FAZENDA, 2016).

Este trabalho pretende contribuir para esse projeto, fazendo uso de uma rede bayesiana para a classificação dos dados de entrada, obtidos por meio de sensores, que permita chegar a conclusões com relação ao nível de CS do usuário de dispositivo móvel.

Uma rede bayesiana (RB) é um modelo probabilístico, na forma de um grafo acíclico dirigido, construída com nodos, representando variáveis aleatórias discretas ou

contínuas, e arcos direcionados, representando as relações entre elas. Os nodos filhos são dependentes de seus pais, e possuem uma probabilidade associada. As RB's são construídas usando conhecimento de especialistas ou usando algoritmos eficientes que aprendem a rede (BUCZAK, 2016).

Dependendo da aplicação, a rede pode ser usada para explicar a interação entre as variáveis ou para calcular um resultado provável com base nos dados de entrada. As tabelas de probabilidade podem ser calculadas a partir dos dados de treinamento disponíveis. Entre os principais objetivos para o treinamento de RB's estão inferir variáveis não observadas, a aprendizagem de parâmetros e a aprendizagem da estrutura (BUCZAK, 2016).

Para que o aplicativo conseguisse chegar a uma conclusão quanto ao nível de consciência do usuário, poderia-se realizar a inferência bayesiana em um servidor remoto ou de dentro da própria aplicação. Para o presente trabalho procurou-se seguir essa última alternativa, adaptando e fazendo uso de uma API desenvolvida na linguagem Java mas adaptada para ser executada em uma aplicação Android.

## 2. Motivação

O que motivou este trabalho foi a vontade de criar uma ferramenta que ajudasse os deficientes visuais a se contextualizarem-se com o ambiente, pelo que seria desenvolvido um aplicativo para dispositivos móveis que fizesse a análise do ambiente através de sensores, como câmera e microfone, classificando pessoas e objetos, e fazendo conclusões quanto ao lugar em que o indivíduo se encontrasse. Seria necessário a utilização de aprendizado de máquina para implementar a classificação.

Mas este autor teve contato com outro projeto, intitulado "Computational Model of Situational Awareness for users of smartphones in the vicinity of traffic", que trata da percepção do ambiente e desatenção por parte do usuário de dispositivos móveis, com foco em situações em que o colocariam em perigo, sendo que a utilização de aprendizado de máquina nesse caso seria muito semelhante com a ideia do aplicativo para deficientes visuais. Dessa forma, surgiu a oportunidade de utilizar os dados coletados no projeto para o desenvolvimento de um modelo de aprendizado de máquina, o que pouparia o trabalho de coletar dados, mas que implicaria em desenvolver um modelo voltado para o projeto. Optou-se por seguir esse caminho. Apesar disso, o modelo ainda pode ser aproveitado para ambas as ideias.

Com o modelo desenvolvido seria possível utilizá-lo em um aplicativo para dispositivos móveis conforme descrito no início desta motivação.

## 3. Fundamentação Teórica

### 2.1 Consciência Situacional

Endsley (1995) define consciência situacional (CS) como "a percepção de elementos do ambiente considerando tempo e espaço, a compreensão de seu significado e a projeção de seu status em um futuro próximo". Também pode ser considerada um campo de estudo preocupado em compreender os ambientes em que é crítica a tomada de decisão, como acontece em áreas complexas e dinâmicas como aviação, controle de tráfego aéreo e operação de grandes e complexos sistemas, ou mesmo para tarefas mais comuns do dia-a-dia como caminhar e dirigir. Assim, alguém com um bom nível de CS tem grande chance de tomar decisões corretas em sistemas dinâmicos e complexos. Além disso, o estudo e entendimento dos componentes que constituem a CS possibilitam o

desenvolvimento de interfaces mais eficientes e de melhores programas de suporte à decisão (ENDSLEY, 1995).

O presente trabalho tem como foco o impacto que o uso de dispositivos móveis tem na CS. O uso de smartphone afeta a percepção do usuário e ao tentar utilizar esses dispositivos enquanto realiza alguma outra tarefa, como dirigir, andar de bicicleta ou mesmo caminhar perto de rodovias e ferrovias, têm grande probabilidade de causar acidentes. O uso desses dispositivos afeta a atenção do usuário em dois níveis: oclusão, e cegueira e surdez por desatenção.

A oclusão ocorre quando o usuário não consegue realmente ver ou escutar outros elementos do ambiente enquanto está utilizando o dispositivo. A cegueira e surdez por desatenção acontece quando o usuário, mesmo vendo e escutando outros elementos do ambiente, os ignora devido a sua atenção estar voltada exclusivamente para o dispositivo, mesmo que esses elementos externos coloquem em risco a sua segurança (SANTOS; FAZENDA, 2016).

## 2.2 Aprendizado de Máquina

Lobato (2016) define o aprendizado de máquina como “a habilidade de um computador aprender sem ter sido explicitamente programado para resolver determinada tarefa específica”, sendo o resultado da utilização de algoritmos eficientes que conseguem encontrar padrões a partir de dados disponíveis.

Segundo Buczak (2016), o aprendizado de máquina geralmente consiste de duas fases: treinamento e testes, em que são realizados os seguintes passos:

- Identificar características nos dados;
- Identificar um subconjunto dos atributos necessários para a classificação;
- Aprender o modelo usando dados de treinamento;
- Usar o modelo treinado para classificar os dados desconhecidos.

O aprendizado de máquina pode ser dividido, de maneira geral, em aprendizado supervisionado e não supervisionado. No aprendizado supervisionado, os dados de treino são etiquetados e o objetivo é encontrar uma relação ou modelo que explique os dados. No aprendizado não supervisionado não há etiquetação dos dados e o objetivo nesse caso é encontrar padrões ou conhecimento nos dados (LOBATO, 2016).

## 2.3 Redes Bayesianas

De acordo com Charniak (1991), existem alguns tipos de problemas que queremos que os sistemas resolvam, mas que são difíceis de solucioná-los somente pela lógica. São problemas que envolvem incerteza.

Uma RB pode ser definida como uma rede probabilística, constituída de uma estrutura associada a uma distribuição de probabilidades. Essa estrutura é representada por meio de grafos direcionados, conectados e acíclicos, compostos de nodos e arcos. Os nodos são círculos e correspondem às variáveis aleatórias, e os arcos são setas e representam a dependência probabilística direta entre duas variáveis (BUCZAK, 2016).

Por fazerem uso de uma estrutura gráfica, as RB’s permitem que se perceba as relações de causa entre as variáveis, sendo sua leitura bastante intuitiva, além de facilitar a interpretação dos resultados (SARABANDO, 2010).

Para especificar a distribuição das probabilidades de uma RB, é necessário definir as probabilidades de todos os nodos-raízes (nodos que não possuem predecessor) e as probabilidades condicionais de todos os demais nodos, com base na relação direta

com seu predecessor (CHARNIAK, 1991). Assim, cada variável da rede pode tomar um valor dentro de um intervalo finito e nelas estão definidas as probabilidades condicionais. A probabilidade condicional é a probabilidade de determinado evento X ocorrer sabendo que o evento Y ocorreu. O Teorema de Bayes relaciona as probabilidades dos dois acontecimentos X e Y com as suas probabilidades condicionais mútuas.

Quando uma variável tem valor conhecido e inserido na rede, temos o que se chama de evidência, sendo utilizada para realizar inferência. Inferência é um termo que se refere a atualização das probabilidades da rede com base nas evidências, ou seja, uma vez inserido um valor conhecido na rede, as probabilidades precisam ser ajustadas (ARA-SOUZA, 2010). As redes bayesianas nos permitem calcular as probabilidades condicionais mesmo que tenhamos os valores de apenas alguns nodos. (CHARNIAK, 1991).

Pode-se construir uma RB manualmente, através do conhecimento obtido por meio de um especialista e da literatura, ou pode-se aplicar técnicas de aprendizagem para a estimação dos parâmetros e da estrutura, através de algoritmos aplicados sobre a base de dados (SARABANDO, 2010).

## 4. Desenvolvimento

### 4.1 Estudo de Caso

O projeto Road Awareness se propõe a desenvolver um modelo computacional de CS voltado para usuários de dispositivos móveis que utilizam seus aparelhos próximos de rodovias e ferrovias, com o objetivo de melhor compreender como esses usuários se comportam nesse ambiente e em que situações sua segurança é comprometida. Através dos dados obtidos e do estudo realizado seria possível melhorar a segurança dos pedestres (SANTOS; FAZENDA, 2016).

O projeto pretende caracterizar os níveis de deficiência (oclusão e cegueira por desatenção) quando os dispositivos móveis estão sendo usados e combiná-los com o perfil de comportamento do usuário e recursos coletados pelo próprio dispositivo para construir modelos computacionais que prevejam o nível de CS de um usuário (SANTOS; FAZENDA, 2016).

O projeto faz uso de uma plataforma de realidade virtual 3D, localizada na Universidade de Salford, Reino Unido, em que o usuário é imerso em um ambiente urbano simulado. O ambiente simula uma cidade urbana, contendo vias em que circulam carros, possuindo uma faixa de pedestres em que circulam pequenos animais. Junto à faixa de pedestres fica um semáforo que estabelece a passagem desses animais quando está verde e bloqueando-os quando fica vermelho. Os animais presentes na simulação eventualmente atravessam a faixa, mas ignoram se vem ou não carros pela via. O experimento consiste em um usuário ficar próximo da faixa, sendo responsável por controlar o semáforo por meio de um smartphone, devendo deixar o sinal verde assim que for seguro atravessar pela faixa para que os animais possam fazer isso sem que sejam atropelados. O experimento considerou ainda a possibilidade de utilizar distrações para tirar a atenção do usuário. Assim, em meio às várias atividades que precisa executar, é analisada a CS do usuário.

**Figura 1 – Usuário imerso no ambiente da simulação. Fonte: Pereira (2017)**

## 4.2 Sistema Proposto

O sistema proposto consiste de três componentes: uma entrada com dados de sensores, um processamento interno e uma saída em forma de alerta ao usuário.

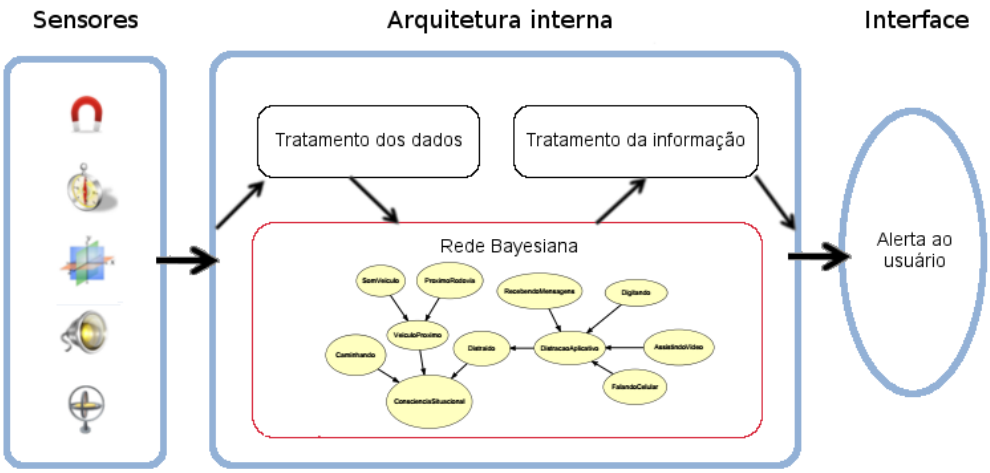


**Figura 2 - Diagrama do sistema proposto. Fonte: Do autor (2017)**

Através da API Android é possível obter dados de sensores, como por exemplo acelerômetro, giroscópio e campo magnético. Através da API também é possível identificar algumas ações do usuário, como ouvir música, se ele está digitando, recebendo mensagens ou falando ao celular.

O processamento interno visa identificar o nível de consciência do usuário, e para isso é utilizada uma RB. Dessa forma, a medida que os dados são obtidos através dos sensores, é realizado o tratamento desses dados para possibilitar a inserção das evidências na rede. O resultado dessa inferência bayesiana representa o nível de CS do usuário em relação ao ambiente em que se encontra, e essa informação é tratada para então ser emitido um alerta ao usuário.

A aplicação executa em background, sempre monitorando o ambiente e a interação do usuário com o dispositivo, e uma vez identificado um nível baixo de CS,

seja pelo uso intenso do dispositivo ou pela identificação de um ambiente mais crítico, ou a composição de ambos, a aplicação emite alertas ao usuário, seja por meio de avisos sonoros, visuais ou por vibração.

### 4.3 Construção da Rede Bayesiana

Pereira (2017), em seu trabalho de conclusão de curso, também utiliza como estudo de caso o projeto Awareness. Em seu trabalho ele constrói uma RB com o fim de desenvolver e experimentar uma técnica de aprendizagem por reforço com o propósito de aprender os parâmetros da rede.

Para a construção da RB, ele utilizou os dados coletados através de um experimento realizado no ambiente virtual, conforme estudo de caso, em que participaram 20 usuários. Além de ter que executar as atividades do simulador, os participantes foram submetidos a três diferentes situações: (1) o participante deveria apenas apertar o botão indicando estar consciente com relação ao carro; (2) ele deveria, além de apertar o botão indicando consciência, também responder perguntas no smartphone; e (3) ele deveria, além de apertar o botão indicando consciência e responder as perguntas, usar ainda fones de ouvido. No simulador os carros poderiam vir de quatro direções, frente, atrás, direita e esquerda, e poderiam vir com ou sem som. O tempo que o participante levou para indicar a consciência também foi considerado. Conseguindo executar as tarefas com sucesso, o participante era considerado como consciente. Com base nos dados, a estrutura da RB é construída de forma manual. Após a aplicação do aprendizado por reforço, obteve-se a rede mostrada na figura:

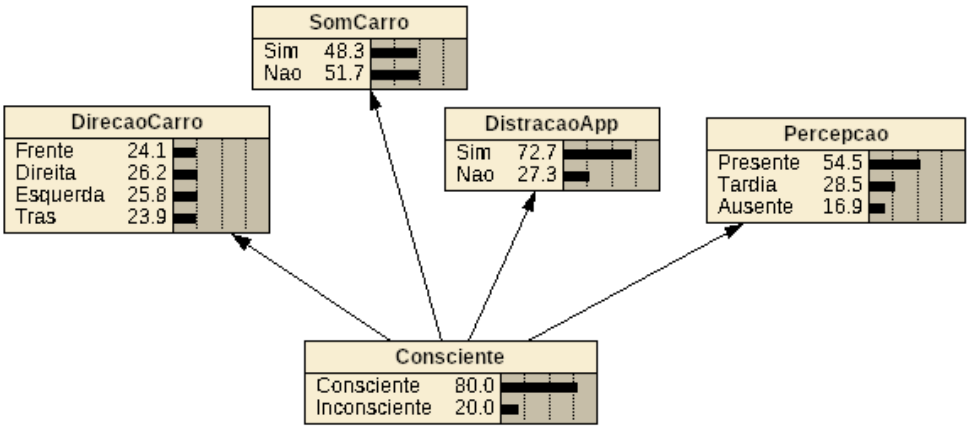


**Figura 3 - RB após aplicação do aprendizado por reforço. Fonte: Pereira (2017)**

Considerando que a intenção do presente trabalho é contribuir para o projeto Awareness, optou-se por utilizar os mesmos nodos com as mesmas variáveis da rede apresentadas por Pereira (2017) para a construção da RB, assim como a especificação das probabilidades.

### 4.4 Aplicativo para Segurança de Pedestres

A fim de que o aplicativo pudesse inferir a CS, seria necessário que ele aceitasse a inserção de evidências e que executasse algum algoritmo de inferência bayesiana dentro do próprio dispositivo. Dentre as opções disponíveis, foi escolhido o UnbBayes, uma vez que essa ferramenta tem o foco em RB, e foi possível compreender bem o projeto através do artigo de Matsumoto (2014), o qual possui boas explicações e bons

exemplos.

### 4.4.1 Adaptação do UnBBayes para Android

UnbBayes é uma aplicação desenvolvida pelo grupo de Inteligência Artificial do departamento de Ciências da Computação da Universidade de Brasília, que tem como premissa possibilitar a construção de modelos gráficos probabilísticos e a realização de inferência. Possui interface gráfica e faz uso de plugins para estender suas funcionalidades (MATSUMOTO, 2014). O projeto foi desenvolvido na linguagem Java e implementa como mecanismo padrão de inferência o algoritmo de junção de árvore (MATSUMOTO, 2014).

Para poder utilizar o UnbBayes na plataforma Android como uma biblioteca, foi necessário realizar algumas adaptações, uma vez que a API do Android, apesar de ser baseada em Java, não processa as classes Java de Interface Gráfica (Java Swing, por exemplo), ou classes que utilizam hardware, uma vez que possui classes específicas. Portanto, foi necessário retirar do projeto essas classes. Foi necessário também retirar a funcionalidade de carregamento de plugins. Também foram retiradas a maioria das dependências de classes externas, como aquelas que tratam de logs, pois muitas delas eram incompatíveis com a API do Android. Os seguintes arquivos foram retirados do projeto original: commons-logging-1.0.4.jar, javahelp-2.0.02.jar, jaxme2-rt-0.5.1.jar, jaxmeapi-0.5.1.jar, jpf-1.5.jar, log4j-1.2.17.jar, xalan-2.7.0.jar e xml-apis-1.0.b2.jar.

O resultado é um projeto leve, com tamanho de 4,74 MB, e o arquivo JAR com tamanho de 1,87 MB, frente aos 48,5 MB do projeto original. Porém possui um único algoritmo de inferência, o de junção de árvore, além de perder várias funcionalidades.

### 4.4.2 Sensores

Dentre os usos frequentes do smartphone, os mais comuns são jogar, assistir música e se comunicar com outra pessoa, seja por fala ou por meio de mensagens de texto. Considerando que o aplicativo será executado em segundo plano, uma vez que a ideia é analisar o uso do smartphone, houveram algumas restrições quanto ao que poderia ser monitorado.

A identificação de jogar, por exemplo, é complexa, uma vez que essa atividade faz uso intenso de diversos recursos do dispositivo. A identificação de toque na tela ou mesmo digitação de caracteres no teclado virtual atualmente é restrita à aplicação que executa essas atividades, não sendo acessíveis por aplicações externas. Em versões anteriores do Android era possível realizar esta identificação de fora da aplicação, mas isso foi bloqueado com versões recentes da API. Além disso, dependendo do modelo do smartphone, ele pode vir muitas vezes com apenas um único sensor, que geralmente é o acelerômetro, o que restringe bastante o desenvolvimento de uma aplicação que consiga abranger uma gama maior de dispositivos.

Portanto, foram testados somente a identificação de caminhada, ouvir som, chamada recebida/iniciada, e tela ligada.

Considerar a identificação da caminhada pelo usuário do smartphone foi uma questão bastante relevante no desenvolvimento do aplicativo, uma vez que quando se fala na segurança do pedestre, dificilmente ele estaria em perigo se estiver parado. Mas não chega a ser uma atividade que pudesse ser classificada como uma distração causada pelo uso do aparelho, e por isso não foi considerada como uma evidência que pudesse ser inserida na RB.

Com relação a identificação de tela ligada, a probabilidade de que o usuário esteja utilizando o smartphone e distraído é muito grande, mas colocá-la como

evidência na rede implicaria que só o fato de ligar o dispositivo o aplicativo classificaria a consciência como baixa, independente de qualquer outra distração. Dessa forma, ela não foi considerada como uma evidência a ser inserida na rede. Outra situação é que o usuário poderia estar distraído mesmo com a tela desligada, como ocorreria na oclusão auditiva ao usar fones de ouvido.

Ouvir música e estar com uma ligação telefônica ativa são atividades que realmente distraem o usuário, e assim foram consideradas como evidências na rede.

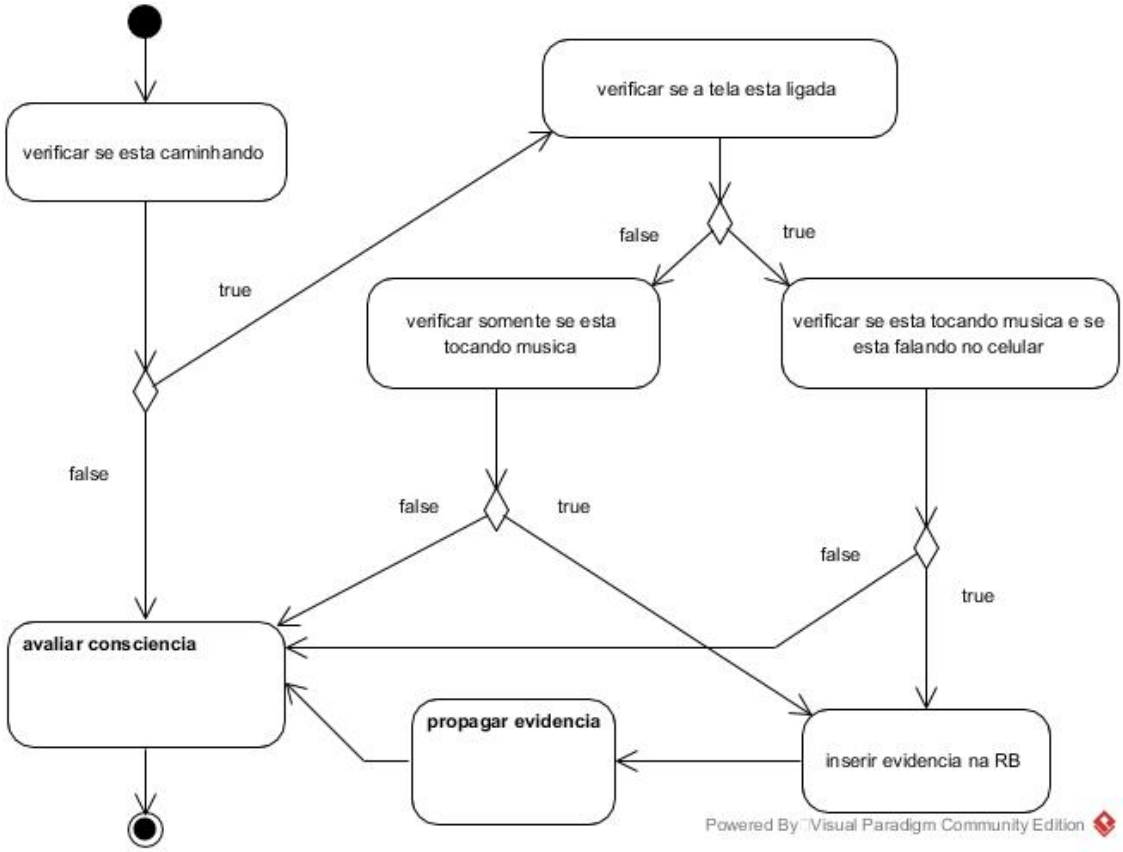


**Figura 4 - Diagrama de atividades do tratamento de dados para inserção na RB. Fonte: Do autor (2018)**

Dessa forma, o tratamento dos dados dos sensores dentro da aplicação, para eventual inserção de evidência na RB, é feito conforme o diagrama da Figura 4:

1. Primeiramente é analisado se o usuário está caminhando, sendo esta uma condição necessária para possível inserção de evidência. Se confirmada a ocorrência de caminhada, passa-se para uma segunda análise. Senão, passa-se para o final, onde é verificado o nível de consciência;
2. Em um segundo momento, havendo caminhada, é feita a verificação de tela ligada, o que apenas altera a verificação dos demais dados dos sensores;
3. Se a tela não está ligada, é verificado apenas se há algum som sendo reproduzido, considerando a situação de escutar música com a tela desligada;
4. Se a tela está ligada, é verificada se há a ocorrência de ligação telefônica ou se algum som está sendo reproduzido. Uma chamada telefônica é identificada após se apertar o botão de telefone, tanto para chamadas efetuadas quanto para chamadas recebidas. Os sons aqui identificados podem ser tanto de reprodução de música quanto originados por players de vídeo;
5. Havendo a identificação de som ou chamada telefônica, é feita a inserção da

evidência na RB e propagada para o restante da rede, finalizando com a avaliação do nível de consciência. Não havendo, passa-se para o final, onde é verificado o nível de consciência.

O modelo desenvolvido no projeto Awareness e utilizado para a construção da RB também trouxe algumas limitações quanto à inserção de evidências na rede. Com relação ao nodo DirecaoCarro, não haveria como o smartphone identificar a direção do carro (esquerda, frente, etc.) em relação ao usuário/aparelho. Também com relação ao nodo Percepcao, seria difícil identificar se a reação do usuário é tardia, presente ou ausente. Na verdade, o propósito do aplicativo é justamente suprir a ausência da percepção. Somente com relação à identificação de som do carro para o nodo SomCarro, acredita-se que haveria a possibilidade de se implementar, pois o tratamento e identificação de sons não é algo novo, mas isso aumentaria demais a complexidade do trabalho, e por isso foi deixado de lado.

### 4.4.3 Implementação do Aplicativo

O aplicativo foi implementado na plataforma Android, utilizando a IDE Android Studio versão 3.1.2., separando-os em três pacotes: view, control e model.

No pacote view foi criada apenas uma classe, PrincipalActivity, responsável por iniciar a aplicação, assim como pela interação com o usuário, e por isso foram criados os métodos mostreInfoRB() e notificar(). O método mostreInfoRB() imprime na tela as informações relacionadas à RB, e o método notificar() realiza a notificação do usuário abrindo um popup na tela com aviso sonoro, além de um ícone permanecer na barra superior do sistema até que o usuário o visualize.

No pacote control, foi criada a classe Controlador, responsável por fazer a intermediação entre a interface do usuário e o domínio do problema. Uma vez que a aplicação é inicializada pela classe PrincipalActivity, ela imediatamente passa o controle para a classe Controlador. Essa classe fica aguardando que a classe ActivityRecognizedService, ou o próprio sistema operacional no caso de ligação telefônica, informe qualquer um dos usos possíveis do smartphone, conforme definidos neste trabalho, encaminhando os dados obtidos através dos sensores para serem processados na RB. O resultado obtido da inferência bayesiana é então encaminhado para ser mostrado na interface gráfica.

Por fim, no pacote model, foram criadas duas classes: ActivityRecognizedService e RedeBayesiana. Para medir a CS enquanto o usuário está distraído com algum aplicativo do smartphone, foi necessário que a aplicação executasse em background, ou seja, o usuário poderia utilizar qualquer funcionalidade do seu aparelho enquanto a aplicação ficaria monitorando o seu uso. Esse monitoramento é feito utilizando a classe ActivityRecognizedService, que implementa o chamado serviço, que fica constantemente verificando os estados dos sensores. No momento que o serviço identifica uma atividade, ele realiza uma transmissão sinalizando para a aplicação que determinada atividade aconteceu. Apenas com relação a identificação de chamada telefônica, ela não é tratada pela classe ActivityRecognizedService, pois o próprio sistema realiza uma transmissão global na ocorrência de ligação, capturada pelo método onReceive() da classe Controlador.

Já a classe RedeBayesiana faz a implementação da RB, fazendo uso da biblioteca do UnbBayes. Através do método atualizaRB() as evidências são recebidas da classe Controlador, que alimentam a RB e que por sua vez propaga as evidências para o restante da rede. O algoritmo de inferência bayesiana utilizado foi o de junção de

árvore, por ser o único disponível, uma vez que a utilização de outros algoritmos é feita por meio de plugins, que acabaram tendo que ser retirados devido a incompatibilidade com a API do Android. Foi criado o método retornaRB() para, assim que requisitado, retornar os dados da rede para eventual impressão na tela. Foi necessário ainda criar o método criarArquivoInputStream(), para carregar o arquivo que contém a RB. Optou-se por carregar apenas arquivos .net, uma vez que seria o suficiente para os propósitos deste trabalho, devendo o arquivo estar dentro da pasta assets do projeto.

#### 4.4.4 Saída do Sistema

A saída do sistema é composta pelos alertas emitidos ao usuário e por informações que são impressas na interface gráfica, que ocorrem após o processamento interno dos dados. O alerta pode ser uma mensagem de texto que aparece na parte inferior da tela por cerca de 3 segundos, ou pode vir na forma de uma notificação na barra de status que fica na parte superior da tela, que se apresenta juntamente com um popup de mensagem e um aviso sonoro, além de fazer vibrar o aparelho.

Os avisos são mostrados em dois momentos: (1) quando é identificada a caminhada juntamente com alguma distração; e (2) quando o nível de consciência é considerado baixo. Na primeira situação, é impressa na interface gráfica informações relacionadas à RB e qual tipo de distração ocorreu. Além disso, ocorre a notificação do usuário por meio da mensagem "Atenção! Caminhar e utilizar o smartphone não é seguro!". No segundo momento, é avaliada a consciência. Não havendo evidência inserida na rede, ela apresenta a probabilidade de 80% para Consciente. Por outro lado, havendo a inserção de evidência no nodo DistracaoApp para a ocorrência de distração, a probabilidade cai para 77,1% para Consciente. Dessa forma, o alerta ao usuário acontece quando o resultado da RB retornar um valor menor que 79%, feito por meio da mensagem "Nível de consciência abaixo de 0.79!", e uma notificação com o aviso "Atenção! Você pode estar distraído!".

### 4.5 Testes

Os testes foram realizados apenas com relação ao funcionamento do aplicativo, ou seja, se o aplicativo conseguia identificar o uso do smartphone com base em dados dos sensores e realizar o processamento adequado, mostrando no final informações ao usuário. Não foram realizados testes em ambientes urbanos, como estradas e ferrovias, tampouco para medir a eficácia do modelo em inferir a CS.

Para os testes foi utilizado um smartphone da marca Motorola, modelo Moto C Plus, rodando o Android na versão 7.0, equipado apenas com GPS e acelerômetro, não possuindo bússola, giroscópio e sensor de proximidade. A área de um corredor com cerca de 9 metros de comprimento foi suficiente para os testes.

Foram realizados testes preliminares de cada sensor, analisando-os separadamente:

- Uma vez inicializada a aplicação, a identificação da caminhada teve tempos de resposta entre 2 e 30 segundos. Não foi possível identificar o motivo de isso acontecer, provavelmente deve estar ligado à prioridade que o sistema concede ao serviços que rodam em background. Também pode estar relacionado com a maneira que a API caracteriza a caminhada, uma vez que pode-se caminhar rápida ou lentamente, segurando o dispositivo firme ou de forma relaxada;
- A identificação de tela ligada aconteceu de modo satisfatório, com resposta imediata;

- A identificação de sons tocando, uma vez inicializada a aplicação, ocorreu com tempos de resposta entre 3 e 15 segundos, provavelmente também devido à prioridade concedida pelo sistema. O aplicativo consegue identificar sons gerados tanto por um tocador de música como por um player de vídeo. Testes com a visualização de vídeos na internet foram positivos;
- A identificação de chamada telefônica ocorreu de modo satisfatório, tanto para chamadas realizadas como para recebidas, com resposta imediata.

Os testes com a versão final do aplicativo mostraram os mesmos comportamentos dos sensores, mas necessariamente foram realizados utilizando o smartphone e caminhando ao mesmo tempo, o que significa que todos os testes finais tiveram tempos de resposta entre 3 e 30 segundos. Nos testes a tela também deveria estar ligada, com a exceção de escutar música.

Foram também realizados testes com as notificações ao usuário. As notificações se mostraram adequadas pois conseguiam se sobrepor a qualquer aplicação em uso. Por exemplo, ao caminhar e assistir a um vídeo na internet, o aplicativo exibe uma mensagem na tela por cerca de 3 segundos, sobrepondo o vídeo em execução, além de um aviso sonoro que também se sobrepõe ao som gerado pelo vídeo, fazendo ainda o aparelho vibrar e mostrando um ícone na barra superior da tela.

## 5. Conclusão

Este trabalho se propôs a desenvolver um aplicativo que fizesse uso de RB, o que foi alcançado com a utilização da API do UnbBayes. Era essencial que a inserção de evidências e inferência bayesiana fosse processada dentro da aplicação em tempo real, pois em uma situação extrema a resposta precisa ser imediata, considerando que o que está em jogo é a segurança do usuário. O processamento da inferência bayesiana em um servidor remoto nesse caso seria inviável, pois poderia ocorrer uma falha na comunicação ou mesmo uma demora na resposta. O uso do UnbBayes se mostrou adequado, sendo esta uma ferramenta open source, mostrando alta compatibilidade com o sistema Android.

Uma vez que ainda são poucas as ferramentas que permitem a utilização de IA em aplicativos móveis, a adaptação da API do UnbBayes possibilitando o uso de RB em uma aplicação Android pode ser considerada uma contribuição deste trabalho para o desenvolvimento de aplicativos que trazem a IA para dentro do dispositivo.

Este trabalho também propôs que o aplicativo fosse desenvolvido para a segurança dos pedestres, o que foi alcançado em parte, tendo em vista as limitações impostas pela plataforma e pelo modelo. Em uma situação ideal a implementação de um mecanismo de segurança no smartphone deveria ser feito pela proprietária do sistema operacional, considerando a necessidade da aplicação rodar em background e a dependência de acesso a hardware. Mas ainda assim, o aplicativo consegue analisar de modo satisfatório o uso do aparelho por meio de dados dos sensores, processando esses dados e chegando à conclusões com relação ao nível de CS, notificando o usuário por meio de avisos visuais e sonoros.

Apesar do modelo de CS não considerar a caminhada, foi percebido durante os testes que faria mais sentido que o aplicativo fizesse a notificação do usuário somente na ocorrência de caminhada, pois seria a situação real de risco, pois do contrário o usuário seria incomodado com as notificações mesmo nas situações normais de uso do aparelho.

Uma das motivações do trabalho foi contribuir com o projeto Awareness, e

acredita-se que isso tenha sido alcançado com a construção do aplicativo e do uso do modelo de consciência situacional que foi desenvolvido no projeto. É importante destacar que houveram algumas limitações do uso do modelo, pois percebe-se que o mesmo foi construído com base na problemática estudada, que levou em conta questões como a oclusão e a percepção do ponto de vista do usuário, mas questões essas que seriam difíceis de se detectar somente com os sensores. Talvez em um futuro próximo seja possível identificar a direção de qualquer veículo em relação ao dispositivo móvel.

Dessa forma, conclui-se que todos os objetivos foram alcançados.

## 6. Referências

ARA-SOUZA, Anderson Luiz. REDES BAYESIANAS: UMA INTRODUÇÃO APLICADA A CREDIT SCORING. 2010. 99 f. TCC (Graduação) - Curso de Estatística, Universidade Federal de São Carlos – Ufscar, São Carlos, 2010.

BUCZAK, Anna L.; GUVEN, Erhan. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. Ieee Communications Surveys & Tutorials. p. 1153-1176. julho-dezembro. 2016.

CHARNIAK, Eugene. Bayesian Networks without Tears. Ai Magazine. Menlo Park, (USA), p. 50-63. winter 1991. Disponível em: <https://www.aaai.org/ojs/index.php/aimagazine/article/view/918/836>. Acesso em: 28 jun. 2017.

ENDSLEY, Mica R.. Toward a Theory of Situation Awareness in Dynamic Systems. Human Factors: The Journal of the Human Factors and Ergonomics Society, [s.l.], v. 37, n. 1, p.32-64, 1 mar. 1995. SAGE Publications. http://dx.doi.org/10.1518/001872095779049543.

LOBATO, Antonio Gonzalez Pastana; ANDREONI LOPEZ, M.; DUARTE, O. C. M. B. Um sistema acurado de detecção de ameaças em tempo real por processamento de fluxos. XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC, 2016.

MATSUMOTO, Shou et al. UnBBayes: a java framework for probabilistic models in AI. Java in academia and research, p. 34, 2011.

PEREIRA, Rodolfo Lottin. Aplicação de Aprendizagem por Reforço para um Modelo Bayesiano de Consciência Situacional. 2017. 119f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade Federal de Santa Catarina – UFSC, Florianópolis, 2017.

SANTOS, Elder Rizzon; FAZENDA, B. M.. Computational Model of Situational Awareness for users of smartphones in the vicinity of traffic. [S.l.], 2016.

SARABANDO, Ana Cristina Lopes. Um estudo do comportamento de Redes Bayesianas no prognóstico da sobrevivência no cancro da próstata. 2010. 84 f. Dissertação (Mestrado) - Curso de Medicina, Universidade do Porto, Porto, Portugal, 2010. Disponível em: <https://repositorio-aberto.up.pt/bitstream/10216/55452/2/tesedo MIMVF.pdf>. Acesso em: 28 jun. 2017.